НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6. 050201 «Системна інженерія»

на тему: «Чат-бот тестування студентського контингенту кафедри»

Виконав:
студент IV курсу, групи ІА-51
Шибецький Владислав Якович _____

Керівник:
Старший викладач Яланецький В.А.

_____

Рецензент:

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 рік

Пояснювальна записка
до дипломного проекту
на тему: «Чат-бот тестування студентського контингенту кафедри.»

Київ – 2019 рік

Зміст

ВСТУП

Між всіма навчальними закладами як на рівні країни так і на рівні світу постійно йде жорстка конкуренція за титул найкращого освітнього закладу. КПІ відомий на весь світ своїми видатними випускниками та відповідає усім вимогам навчального закладу ищого рівня. Випускників нашого навчального закладу з радістю приймають на роботу не лише в українських а й в більшість іноземних компаній. Також КПІ завжди мав титул провідного технічного вузу країни, а отже ми повинні не лише йти в ногу з часом а й задавати темп. Але у чому полягає ця боротьба і які її метрики? Відповідь на це запитання дуже проста — престиж будь-якого навчального закладу у першу чергу визначається знаннями та здобутками його випускників. Отже, необхідно постійно покращувати навчальний процес, але як це робити і що для цього необхідно?

Запорукою продуктивного навчального процесу є можливість отримання студентом зворотного зв’язку від керівників навчального процесу, а для об’єктивної оцінки успішності студентів виникає постійна необхідність у проведенні тестування. Тестування — одна з основних форм контролюта якості освіти. Саме від можливості проведення тестування студентського контингенту залежить контроль успішності, а отже і якість освіти.

У студентів є безліч можливостей зв’язку з викладачами та керівництвом кафедри, але далеко не всі вони є зручними та ефективними. Те саме можна сказати і про проведення тестувань на кафедрі — реалізовано безліч різних варіантів проведення тестів, але деякі з них, відверто кажучи, давно застаріли. Наприклад, у наш час соромно проводити тестування у провідному технічному вузі країни на листочках. Саме тому виникає необхідність у оптимізації цих процесів та надання свого рішення, що зможе задовольнити потреби як студентів так і викладачів.

Система, що розробляється, повинна давати змогу проводити якісне тестування як мінімум одному потоку студентів, це приблизно 120 студентів, а також витримувати набагато більші навантаження. Також система має бути

гнучкою і надавати можливість для налаштування під конкретну задачу чи навіть модернізацію самої системи. Ще однією з найважливіших задач є створення такої системи, яку можна буде швидко інтегрувати у освітній процес.

Цілями створення цієї системи тестування студентського контингенту кафедри є забезпечення відповідного рівня проведення тестування для можливості об’єктивного оцінювання студентів, що допоможе моніторити процес навчання, розуміти слабкі та сильні сторони студентів та відповідно підлаштовувати програму занять.

1 ПОСТАНОВКА ЗАДАЧІ

Отже задачею цього проекту є створення чат-боту для тестування студентського колективу, а точніше сказати — системи, оскільки для бота такого рівня необхідний сильна бекенд частина. Цією системою повинно користуватись одразу велика кількість студентів — як мінімум, потік зі 120-ти студентів., для цього слід провести тестування навантаження. Також цією системою будуть користуватися викладачі та відповідальні люди з кафедри

Для керування ботом має бути адмінка двох рівнів — на вищому рівні надаються права користувачам, на нижчому — виконується адміністрування.

Викладач повинен мати змогу додавати та редагувати тести.

Студент після проходження тесту має побачити свій результат,а викладач має мати можливість переглянути результати усіх пройдених тестів.

У адміністратора має бути можливість надавати необхідні права користувачам для використання адмін-панелі нижчого рівня.

Також непогано було ь додати декілька додаткових опцій, а саме:

- можливыст зворотнього з’язку та розсилок
- можливість додавання та скачування необхідної літератури
- можливість переглядати статистику використання бота

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

На сьогоднішній день на нашій кафедрі використовується безліч способів проведення тестування студентів та зворотнього зв’язку між студентами та викладачами, але не існує єдиного рішення для обох задач одночасно, саме у цьому і полягає основна ідея створення цього проекту. Пропоную розбити огляд існуючих рішень на такі підпункти

- огляд існуючих рішень тестування студентського контингенту кафедри
- огляд існуючих рішень тестування в інших установах

На нашій кафедрі основним видом тестування є написання студентами контрольних робіт, тобто після завершення певної теми студентам видається завдання і вони виконують його у письмовій формі на папірці. Проміжковий контроль по більшості дисциплін здійснюється за рахунок виконання студентами лабораторних робіт. Такий підхід не можна вважати ефективним, оскільки поточний контроль успішності у такому випадку є дуже умовним.

На одній з дисциплін було запропоновано відмінний від цього варінт поточного контролю успішності — для допуску до виконання лабораторної роботи потрібно було пройти письмовий тест. Але це не є оптимальним рішенням, оскільки займає багато часу викладача на перевірку.

Зворотній зв’язок між студентами і викладачами здійснюється декількома способами, а саме:

- телефонний зв’вязок
- електронна пошта
- безліч груп у Telegram
- інформаційно-телекомунікаційна система «Електронний кампус»

Телефонний зв’язок не є зручним, оскільки викладач може надавати певну консультацію одночасно лише одному студенту. Отже, якщо потрібно донести певну інформацію групі студентів необхідно знаходити контакт

старости групи або відповідальної особи, передавати їй необхідну інформацію і сподіватися у те, що ця особа розповсюдить цю інформацію серед своїх одногрупців. Також це вид зв'язку не є зручним для студентів, оскільки студент задає питання лише одному викладачу чи працівнику кафедри і тому не завжди може отримати своєчасну та точну відповідь. Також до мінусів можна віднести те, що не завжди зручно вести телефонна розмова чи sms-переписку.

Електронна адреса є у кожного, але не кожен має постійний доступ до своєї скриньки. Так, у багатьох є додато на смартфоні, але не у всіх. Здебільшого людин перевіряє свою поштову скриньку 1-2 рази на добу, через що отримання зворотнього зв'язку займає порівняно багато часу.

У Telegram створено декілька груп — група кафедри та групи по деяким предметам. Перевагою такого зв'язку є те, що інформація передається достатньо великій групі студентів є постійна активність, можна доволі швидко отримати відповіді на певні запитання. Цей спосіб, на мою думку, є найбільш оптимальним серед усіх.

Інформаційно-телекомунікаційна система «Електронний кампус» – прикладне програмне забезпечення, яке є елементом інформаційно-телекомунікаційного середовища університету та використовується для інформаційної підтримки повсякденної діяльності студентів, викладачів, співробітників університету, а так само для інформаційної підтримки всіх видів інноваційної діяльності в університеті (далі ІТС ЕК).

ІТС ЕК об'єднує внутрішні інформаційні ресурси (навчальні, методичні та інші), надає централізований доступ до них на основі єдиних системних і технологічних рішень та забезпечує їх використання для ефективного управління та планування науково-освітнім процесом. Але при всіх плюсах кампус не є ідеальним рішенням, оскільки є дуже складним і незрозумілим для більшості студентів. Потрібно постійно пам'ятати свої логін та пароль, потрібен час щоб зайти та розібратися у ньому. Саме великий набор

функціоналу і рішень, які ця система пропонує робить її складною та нестійкою.

Проблема тестування і зворотнього за’язку з студентським колективом постає не лиш у нашому вузі а й в багатьох інших. Пропоную розглянути декілька таких рішень.

2.1 Система тестування знань Тесторіум

Тесторіум представляє собою безкоштовну систему для створення тестів та проведення різних видів тестування. Система була створена для полегшення роботи викладачів, пов’язаної з проведенням тестування, а також надання можливості студентам проводити самоконтроль та перевірку знань.

Тесторіум пропонує обширний спектр можливостей для викладача, а саме:

- створення тестів
- надання доступу групі осіб до відповідних тестів
- вмвід результатів проходження тестів учнями
- збір статистичних данних по проходженню тетсі учнями
- можливість використовувати готові шаблони тестів
- надання своїх тестів як шаблонів іншим викладачам.

До основних плюсів цієї системи можна віднести те, що вона абсолютно безкоштовна, хоча і потребує реєстрації, вона не потребує ніякого розгортання, оскільки це вже готове і запропоноване рішення реалізоване як вебдодаток.

До основних мінусів потрібно віднести те, що цю систему можна назвати “дерев’яною”, оскільки вона не дає жодної можливості для налаштування під сіої потреби, вона лише надає шаблони. Потрібно постійно авторизуватися на веб-сторінці. Для зручного проходження тестів необхідно використовувати комп’ютер.

2.2 Система тестування INDIGO

Також у переліку систем тестування студентів слід згадати російську розробку — INDIGO. Система тестування INDIGO — це інструмент спрощення процесу тестування і аналізу результатів, що було створено для вирішення наступного спектру задач:

- тестування і контроль знань учнів
- визначення професійного рівня співробітників
- проведення психологічного тестування
- проведення опитувань
- організації олімпіад і конкурсів

Якщо оглянути на структуру цієї системи, то вона складається з адмінки, що представлена програмним клієнтом та користувацької частини — через браузер.

Порівняно з попередньою системою ця вигідно відрізняється тим, що легко піддається модернізації та налаштування під свої конкретні потреби.

Але також слід звернути увагу і на очевидні мінуси, а саме — ця система також розрахована на проходження тестів на комп'ютері, має малий функціонал та не дуже зручний інтерфейс.

2.3 Система контролю самонавчання THINK40

Так, це не зовсім система тестування, але я вважаю доречним навести її у приклад, оскільки моніторингом контингенту займаються не лише навчальні заклади а й багато інших установ. Все більше компаній починає проводити внутрішній моніторинг та тестування своїх співробітників. Це допомагаэє постійно підтримувати співробітників на необхідному професійному рівні та розвивати їх.

Серед таких систем можна навести систему моніторингу у IBM. Вона опирається на THINK40 — це система, яка допомагає співробітнику та

відповідальному за нього менеджеру відслідковувати та планувати своє навчання та прогрес. Ідея полягає у тому, що кожен співробітник витрачає хоча б 40 годин за півроку і це вже непоганий крок на шляху до самовдосконалення. Також співробітникам час від часу приходить запрошеня пройти той чи інший обов'язковий тест або прослухати певний курс для отримання корисних знань. Наприклад, $1^{12}= 1$, а $1.1^{12} = 3{,}13842837672$, і саме тому проходження такого здавалося б дуже простого і, здавалося б, нетривалого обовц'язкового сомонавчання є гарнтом постійного вдосконалення працівником своїх вмінь т знань.

THINK40 надає можливість завантажувати необхідні матеріали чи тести, та розсилати своїм підопічним посилання на необхідний матеріали чи тести.

Цьому прикладу слідує все більше і більше компаній. При тому серед них зустрічаються як великі корпорації так і здавалося б не дуже великі і ніяк не пов'язані з програмною розробкою компаніїю.

Доказом цього також є те, що до фпілансерів постійно надходять замовлення на подібне програмне забезпечення. Наприклад, на минулому місці роботи мені довелося зіткнутися одразу з двома подібними замовленнями на програмне забезпечення для навчання співробітників. Основним звданням цих проектів було надання новим співробітникам можливості у вільний від роботи час ознайомляватися з історією компанії, її продукцією, принципів та методів роботи з клієнтами а також надання можливості керівникам відслідковувати пройдений матеріал. Усе це було реалізовано у вигляді інтерактивного тест-боту який час від часу міг запропонувати прочитати певний розділ для ознайомлення, проходження певного тесту у вигляді інтерактивної гри або змагання з іншим співробітником. Таож до можливостей боту входила можливість відповідати на деякі найбільш часті запитання.

2.4 ЗНО математика: BOT Challenge

Фахівці Харківського національного економічного університету ім. С. Кузнеця розробили чат-бот, який допомагатиме в підготовці до ЗНО з математики. Про це повідомляє МОН

За ствердженням розробників бот допоможе підготуватися до ЗНО за 30 днів. Отримати доступ до бота можна увівши у пошук у Telegram такий запит: HNEU_ZNO_math_bot. Найближчим часом у ХНЕУ обіцяють презентувати боти й з інших предметів.

Основним мінусом цього рішення є те, що це чат-бот для підготовки до ЗНО і його не вийде використовувати для своїх цілей.

Але зараз цікаві лише його основні переваги — зручнмй інтерфейс відсутність зайвих функцій, можливість проходити тести у дорозі з мобільного девайсу.

Висновок до розділу № 2

У ході дослідження аналогів було з’ясовано, яким критеріям повинна відповідати дана система:

- просте розгортання системи
- зручні інтерфейси
- можливість проходження тестів як з ПК так і з мобільного
- можливість внесення змін та покращень
- стійкість

3 Розробка структур моделей та їх реалізація

Структуру моделі краще розглядати з розгорнутою діаграмою розгортання, оскільки на ній добре зрозуміло відображена сама структура та взаємодія між її компонентами. У цьому розділі буде більш детально пояснена ця модель та причини вибору тих чи інших рішень.

Найпростішою структурною одиницею проекту є реалізація самого бота та адмінок до нього. Це основний елемент системи, у якому прописан усі взаємодії та взаємозв'язки. У розділі реалізації саме на цьому елементі буде загострена основна увага.

Основним покращенням для цієї моделі стало створення сендера. Це мікросервіс, на який направляються усі запити. Його основною ідеєю є зменшення навантаження та збільшення стійкості усієї системи. Основною проблемою подібних сиситем є те, що вони валяться від невеликої кількості запитів. Відбувається забиття черги, відмови по таймаутам та «падіння» самої системи. Цю проблему було вирішено реалізацією цього сервісу. Він контролює потік повідомлень, виставляє чергу та виконує пріоритизацію. Навіть якщо на саму систему впаде безмежна кількість повідомлень, сендер прийме їх на себе та почне обробляти. В результаті може «впасти» тільки сендер, а сама система продовжуватиме виконувати інші функції, не пов'язані з повідомленнями. Це може бути перегляд результатів тестування та інше.

Також цікавим рішенням було використання одразу і MongoDB,і SQLite. Причиною цього стало те, що для мікросервісу потібна була швидка і гнучка модель даних, а для самої системи найкраще підходила надійна та чітка.

Висновок до розділу № 3

В результаті розробки було створено систему, що якнайкраще вілдповість на поставлену задачу а також було впроваджено цікаве архітектурне рішення.

4 Вибір засобів та технологій

4.1 Мова програмування Python

Для бекенд-розробки є два найбільш популярних рішення — це PHP і Python. Незважаючи на безперечну популярність PHP, в основі цього проекту було використано Python, а саме Python 3.6. Основними причинами вибору Python стало:

- структурованість та читабельність коду

Це дуже важливо, оскільки після впровадження системи потрібно дати можливість користувачам модифікувати її та змінювати під свої конкретні задачі.

- безліч фреймворків

Це також є дуже корисною опцією особливо для можливості модифікувати систему. Також це сильно спростило розробку, на що буде звернено увагу у наступних підпунктах.

4.2 Фреймворк Django

Django — це патонівський фреймворк, який призначений для спрощення бекенд-розробки. Його основний принцип можна описати наступною фразою — “не повторюйся”, а отже, він призначений для прискорення розробки завдяки боротьбі з так званим “індійським кодом”, тобто зменшення лишнього копіпасту.

Django проектувався спеціально для роботи з Apache і з використанням PostgreSQL як системою управління базами даних, але в результаті постійного розвитку Django отримав можливість працювати з іншими системами управління базами даних, а саме з MySQL, SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere і Oracle. Для цього Django використовує власний ORM,

у якому реалізоване описання моделі даних класами Python, і за рахунок цього відбувається генерація схеми бази даних.

Архітектура Django схожа на MVC (Модель-Представлення-Контролер). Контролер звичайної моделі MVC відповідає рівню Представлення (View) у Django, а логіка Уявлення в Django реалізується рівнем Шаблонів (Templates). Саме тому рівневу архітектуру Django називають «Модель-Шаблон-Подання» (MTV).

Django надає цілий ряд засобів для пришвидшення розробки веб-сайтів. Наприклад, дуже популярним є створення адмін-панелі з допомогою Django. Завдяки цьому фпеймворку можна замість розробки панелі керування можна використовувати вже вбудований додаток, який легко можна включити в будь-який сайт, зроблений за допомогою Django, і навіть є можливість управляти одразу декількома ресурсами з однієї панелі. Цей додаток дозволяє створювати та змінювати будь-який контент сайту, записувати всі дії, і надає зруний інтерфейс для керування користувачами і групами прав.

Цей фреймворк використовується у безлічі великих і відомих сайтів, таких як Instagram, Disqus, Mozilla, The Washington Times, Pinterest, lamoda і ін.

Деякі можливості Django:

- можливість підключення до БД з можливістю підтримки транзакцій
- вбудований шаблон панелі адміністрування
- зручна та розширювана система різних шаблонів
- вбудоване кешування
- вбудовані шаблони функцийц
- можливість підключення різних зовнішніх модулів аутентифікації, а саме LDAP та OpenID.
- вшита бібліотека для полегшення роботи з формами (успадкування, побудова форм по вже існуючой моделі БД)
- можливість автоматичної документація по тегам шаблонів.

Плюси Django:

- швидкість

Django дуже сильно спрощує життя розробнику, оскільки надає готове та перевірене рішення, що значтно економить час та сили, зменшує вірогідність допущення помилок розробником та надає швидке і, можна сказати, оптимальне рішення.

- безпека

Django одразу пропонує використати готове рішення, яке вже має доволі хорошу систему захисту. Це надійне та стабільне рішення “з обгортки”.

- масштабованість

Фреймворк Django пристосований до роботи з високими трафіками, через що отримав свою популярність.

- різнобічність

Django підходить до великого спектру задач і його можна застосовувати для створення майже будь-якого сайту. Це набір шаблонів, компонуючи які можна отримати швидке та оптимальне рішення майже для будь-якої системи.

Мінуси Django:

- Django занадто монолітний

Під час роботи була виявлена одна дуже неприємна особливість — хоча це набор шаблонів, які можн компонувати я потрібно для рішення поставленої задачі, але це все ж таки доволі “твердий” конструктор. Так, його можна використати як захочеться, але не старайся, не вийде з кубиків зібрати ідеальний шар.

Висновок:

Існує дуже багато плюсів і мінусів Django, що не згаданих у цій статті, однак, коли розглядається проект з дедлайном, використання Django для вирішення поставленого завдання - грамотне рішення.

4.3 Фреймворк Flask

Flask, як і Django, — дуже популярний пайтонівський фреймворк. Коли при проектуванні цього проекту поставав вибір між Flask і Django для реалізації цього проекту було прийняте ршення застосувати обидва фреймворки і зараз пропоную розглянути чому було прийняте саме таке рішення.

Flask надає простоту та гнучкість, дозволяючи розробнику самому вибирати реалізацію тих чи інші речей.

Django — це одразу “повний фарш” з купою вже реалізованих і перевірених шаблонів. Так, це зруно, але не завжди доцільно.

Саме тому доцільно було використовувати Django для реалізації адмінки, а Flask для реалізації мікросрвісів. Він надає можливіст зробити просте, але оптимальне рішення, яке не буде перевантажувати систему і буде одразу зроблене під конкретну задачу

4.4 Система керування базами даних MongoDB

Протягом останніх років популярність open source баз даних постійно зростає, особливо ц помітно у порівнянні з комерційними.

Для багатьох додатків, включаючи і цей проект, використовують кілька баз даних для того, щоб задіяти їх сильні сторони. Жодна база даних не може якісно вирішити одразу всі поставлені, саме тому у даному проекті також буде використовуватися SQLite. Пропоную детально розібратися усвідмінностях між MongoDB та SQL-івсикими системами управляння базами даних

Підходи до архітектури:

Для SQL систем характерне використання основного операційного сховища і деяких додаткових сервісів..

Інший підхід у створенні майже для кожного мікросервісу своєї бази даних, що буде створена і оптимізована спеціально під нього. Саме через це

MongoDB часто використовують як додаткову систему управління базами даних.

Чому ж тоді здійснюється порівняння MySQL і MongoDB? Основною причиною є те, що це дві найбільш популярні системи управління базами даних. MySQL — найбільш популярна серед реляційна баз даних, а MongoDB - серед нереляційних база даних. Але не зважаючи на цей факт, це скоріше не конкуренти, а хороше доповнення одна одній.

Для рішення більшості задач підходять обидві системи, але різниця полягає у простоті реалізації та оптимальності принятого рішення. Найчастіше люди використовують лише одну систему, до якої звикли, що одразу робить рішення проблеми неоптимальним. Слід розуміти у чому переваги та недоліки обох систем щоб правильно їх використовувати.

MySQL - це перевірена система, що включає в себе підтримуку складни-запитів з включаючи аналітикою, через що чудово підходить для рішення більш складних задач

Перевагою MongoDB.є використання гнучкий JSON-формата документів. Основною ідеєю є те, що прості запити набагато рідше створюють серйозні проблеми чи затримки. Якщо звернути увагу на питання продуктивності, то MongoDB є оптимальним рішенням для створення простих запитів. Перевагою MongoDBтакож можна вважати гнучкість завдяки цьому формаиу документів.

Другий важливий момент для порівняння — це схема даних. MongoDB добре підходить для створення простиї додатків з коротким циклом життяМодель даних дуже сильно залежить від програми і досвіду команди. Було б дивним сказати, що у нас реляційний або нереляційних підхід до баз даних краще і краще завжди. MySQL — це реляційна база даних, а отже — це добре організовані таблиці та зв’язки, які довго та чітко служитимуть, що добре підходить для важливих структурованих даних, але такий підхід не завжди підходить, оскільки далеко не всі дані добре лягають на таблиці.

У MongoDB структура даних відрізняється тим, що заснована на документах що чудово підходить для зберігання простих даних.

До реляційних даних використовується мова SQL, а в MongoDB і більшості NoSQL-базах даних використовується стандарт CRUD, який говорить, що є операції для створення, читання, видалення і оновлення документів.

MySQL, підтримує транзакції довільного розміру. MongoDB не підтримує транзакції, але він підтримує невеликі, атомарні операції над документом, що ще раз наглядно демонструє та підказує у яких випадках і що краще використовувати.

В MySQL гарна масштабованість в рамках одного вузла а у MongoDB — на багатьох вузлах.

Отже, було вирішено використовувати MongoDB для швидкого мікросервісу — відправки повідомлень.

4.5 Система керування базами даних SQLite

У минулому підпункті була озвучена думка, що неможливо визначити що краще — реляційний чи не реляційна база даних, оскільки кожна з них підходить до рішення конкретної задачі і у великих проектах краще застосовувати обидва підходи. Саме через це у проекті вони застосовані і чудово доповнюють один одного.

SQLite у двух словах можна описати як “мінімальний, але повний набір”. Так, SQLite не підтримує деякі функції, але вбільшості відповідає SQL 92 та вводить свої незовсім стандартні, але зручні особливості.

SQLite працює на надійність, а не на продуктивність, через що її часто критиукують, але незважаючи на це, база даних необхідна для цьлго проекту не є дуже великою чи складною, і тому цей принцип лише грає на руку.

З використанням SQLite була побудована основна частина бази даних.

4.6 Мови розмітки HTML/CSS

Останнім часом набувають популярності різні шаблони для фронтенду, через що все частіше можна почути: “Навіщо вивчати HTML, якщо є безліч простих CMS, де можна створювати сайти збираючи його майже як конструктор — швидко і просто”.

На перший погляд це так. Навіщо винаходити свій велосипед, якщо можна взяти вже готовий і підігнати під себе?. Але , як би дивно це не виглядало, спеціалісти з коасичної верстки досі користуються попитом, який, доречі, постійно тільки зростає. Слід задуматися, чому саме так.

HTML сайти набагато швидше завантажуються ніж CMS системами, оскільки в них немає нічого зайвого

Швидкість завантаження сайту сильно впливає на його популярність. По-перше, це впливає на пошукову видачу, по-друге — жоден користувач не буде чекати поки сторінка, якою б гарною вона б не була, прогрузиться, якщо є альтернатива. Це грає дуже важливу роль особливо в торгівлі. Також HTML сайти дуже мало важать, а отже можна додатково зекономити на.хостингу.

Але у HTML сайтів є свої недоліки, а саме — людський фактор. Творець може допустити помилку, і як завжди, дефект вспливе у самий важливий момент. Усеж таки, набагато краще платити за свою помилку, ніж за зроблену кимось

.Якщо мова йде про HTML, то слід також згадати про CSS, оскільки вони просто створені одне для одного. Дуже складно уявити HTML без CSS і навпаки, оскільки сам по собі HTML — це всього лише розмітка, а за допомогою CSS можна налаштувати стиль. Так, одні з перших сайтів були зроблені лише на HTML, але зараз така картинка може або різати очі, або навіювати ностальгію.

4.7 Мова програмуванняя JavaScript

JavaScript — це прототипно-орієнтована мова програмування. Вона відображає мову ECMAScript, чиїм прототипом спочатку і була. Найчастіше ця мова використовується для надання інтерактивності та сайтам.

За допомогою JavaScript доступні такі функції:

- можливість змінювати сторінки браузерів;
- можливість додавання чи видалення тегів;
- можливість зміни стилів сторінки;
- можливість оримання інформація про дії користувача на сторінці;
- взаємодія з cookie-файлами.

Застосування цієї мови обширне і не обмежен: серед програм, які використовують JS, присутні і тестові редактори, і додатки, і прикладне ПЗ.

Основні переваги JavaScript.

- підтримка усіх браузерів
- проста взаємодія з додатком, що може здійснюється навіть через звичайні текстові редактори.

Webhook

На поточний момент є два способи контролю змін на сторінці телеграм-бота — можна використовувати long polling, або вебхуки..

Принципова відмінність полягає у тому, що при використанні long polling потрібно самостійно запитувати оновлення у API, а використовуючи вебхуки — сервера Telegram відправляють на сервер кожне оновлення за допомогою HTTPS POST-запиту.

Тобто Webhook сильно спрощує процес отримання оновлень. Тепер загрузка йде на

Висновок до розділу № 4

Було розглянуто безліч інструментів для розробки системи, було проведено порівняння та аналіз, в результаті якого було обрано найбільш пыдходящі інструменти для розробки обраної системи. Для реалізації самого бота основним інструментом було обрано Python через величезну кількість плюсів. Одним з них стала величезна кількість фреймворків, які сильно спростили та пришвидшили виконання роботи. Також було прийнято рішення застосовувати як MongoDB так і SQLite, оскільки було прийняте рішення сувмістити плюси обох моделей баз даних.

5 РЕАЛІЗАЦІЯ

5.1 Реалізація функцій

Згадуючи про реалізацію функцій слід обов’язково згадати про основні ласи та алгоритми, які роблять дану систему унікальною.

5.2 Реалізація інтерфейсів

Реалізацію інтерфейсів можна умовно поділити на три складові.

До першої складової слід віднести самого телеграм-бота. Він був реалізований і тримається завдяки Telegram API та Python. Але слід зазначити, що Telegram одразу пропонує рішення для створення бота. В Telegram від початку вшиті зручні інтерфейси для бота а також API, які доступні і завдяки яким можна легко приєднати своє рішення до створеного бота.

До другої складової слід віднести адмін-панель викладача. Вона вцілому була реалізована за допомогою HTML/CSS, саме тому вона має простий та лаконічний дизайн.

Третьою складовою слід вважати панель адміністратора, оскільки вона була повністю виконана завдяки зручному шаблону Django. Так, це доволі просте, але в той же час ефективне і перевірене рішення. Метою цієї роботи було не створення своїх інтерфейсів, а створення системи, яка б була ефективною.

6 Інструкція користувача

6.1 Адмін-панель кафедри

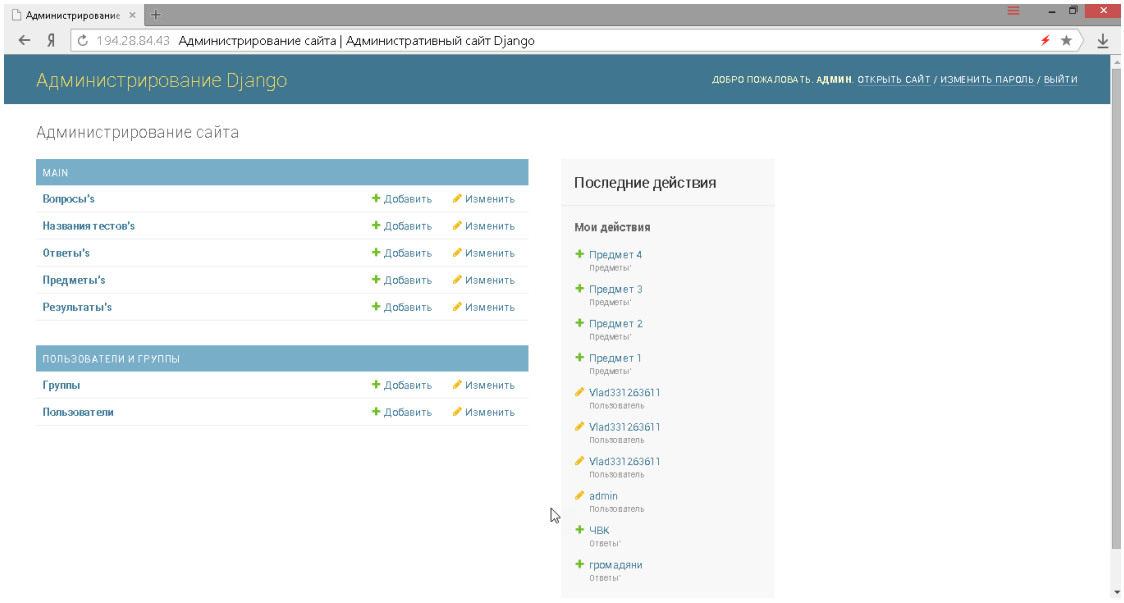


Рисунок 6.1.1 — Початковий екран адмін-панелі кафедри

Адмін-понель кафедри призначена для контролю за роботою усієї системи. Вона реалізована як web-сторінка, що надає можливість додавати та змінювати:

- питання до тестів

На адін-панелі кафедри є можливість створювати та редагувати питання до тестів, але це не є її основною функцією. Ця можливість додана більше для контролю цього процесу та виправлення помилок, ніж для створення нового.

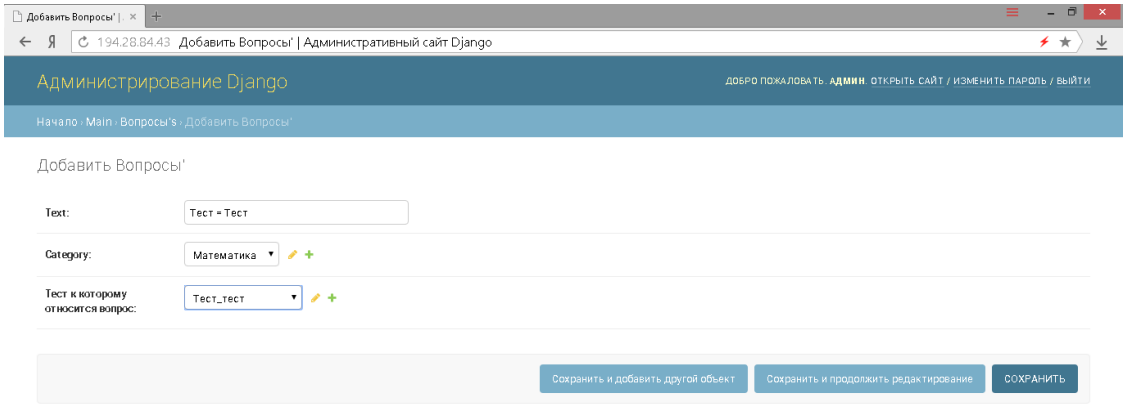


Рисунок 6.1.2 — Екран створення питання до тестів

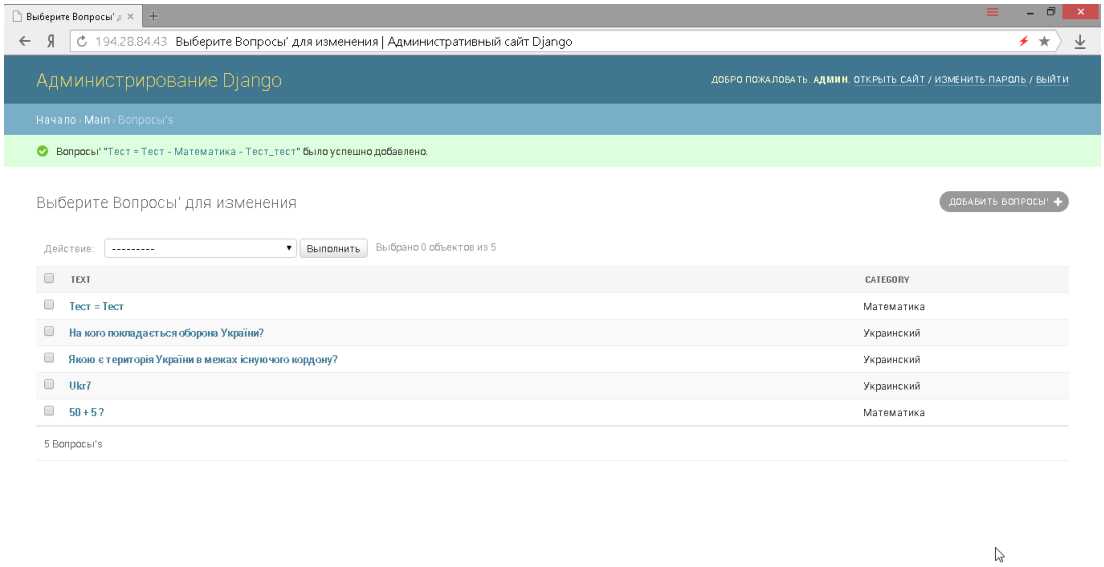


Рисунок 6.1.3 — Екран перегляду питань до тестів

- назви тестів (створювати та змінювати тести)

На адін-панелі кафедри є можливість створювати та редагувати тести, але це також не є її основною функцією і присутня для контролю та виправлення помилок.

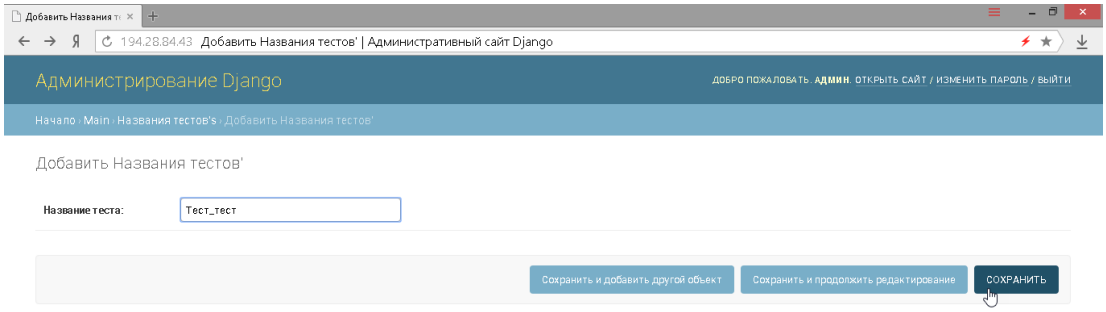


Рисунок 6.1.4 — Екран створення теста

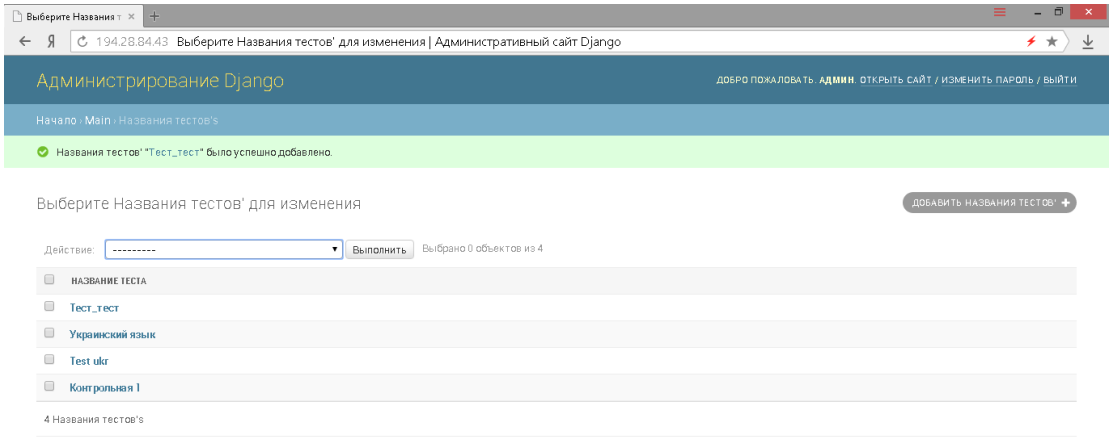


Рисунок 6.1.5 — Екран перегляд тестів

- відповіді

Ще одна функція для керування. Завдяки ній можна створювати варіанти відповідей до питання та обирати вірну відповідь.

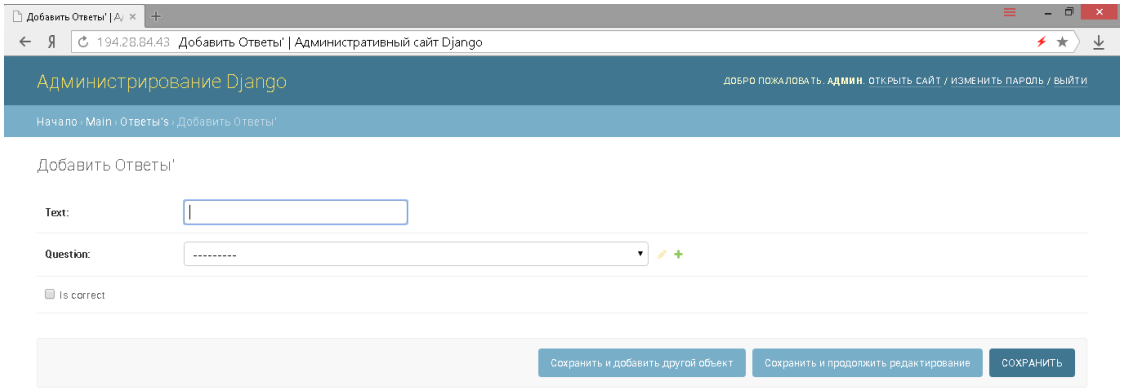


Рисунок 6.1.6 — Екран створення відповідей до тестів

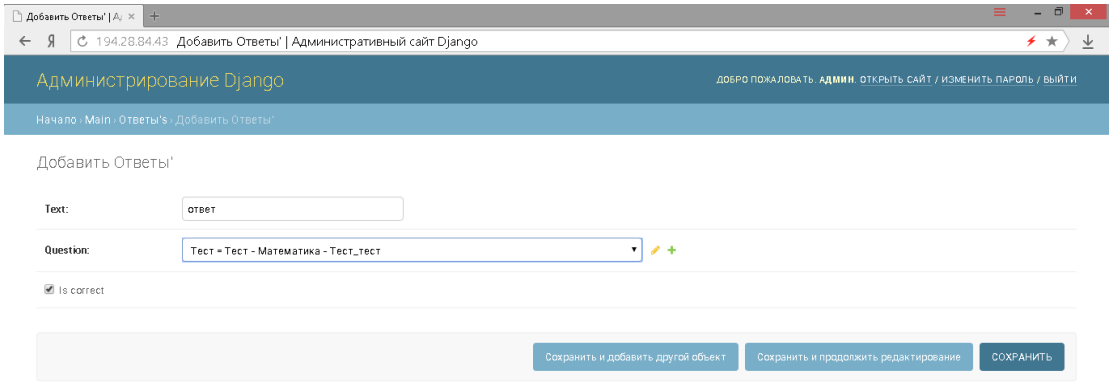


Рисунок 6.1.7 — Екран встановлення правильної відповіді до теста

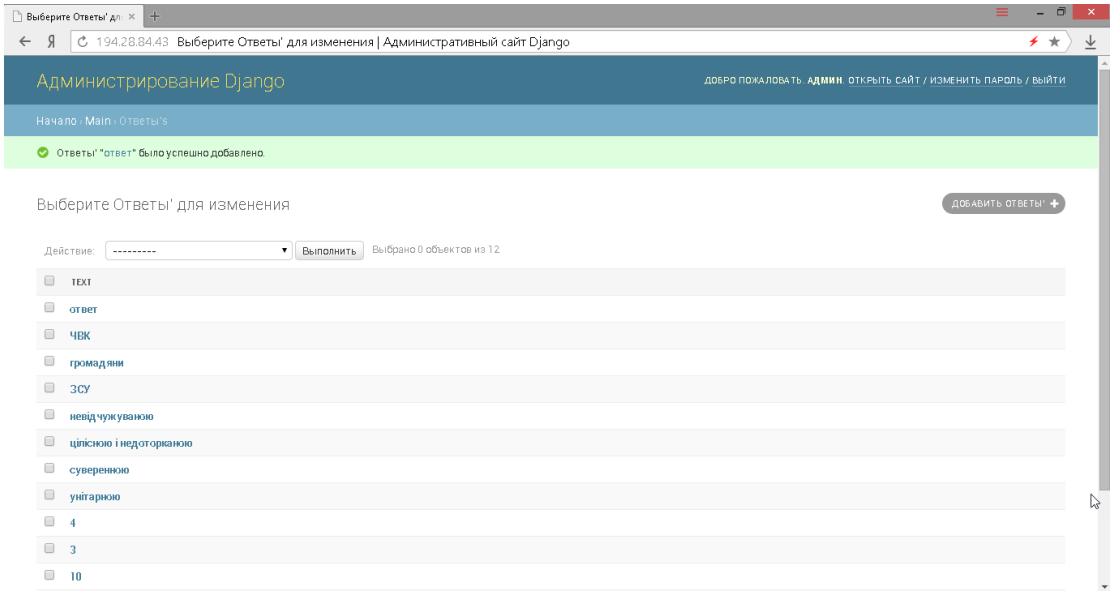


Рисунок 6.1.8 — Екран перегляду відповідей

- предмети

Ця функція доступна лише з адмін-панелі кафедри і дозволяє додавати предмети, до яких згодом можна прикріпити тест чи необхідну літературу.

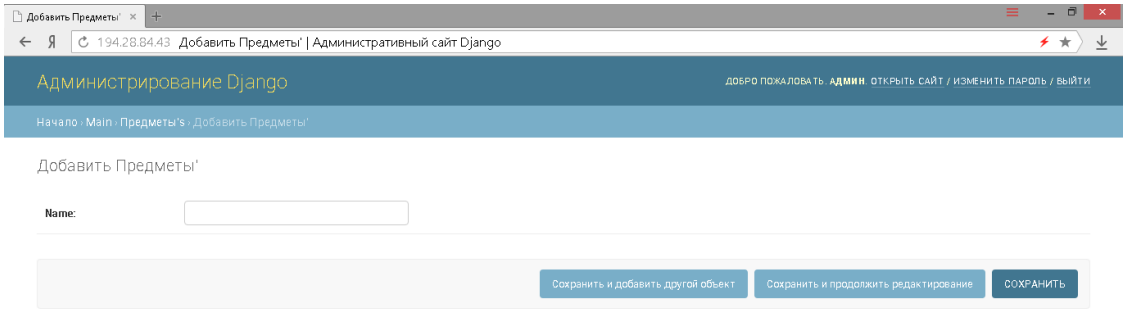
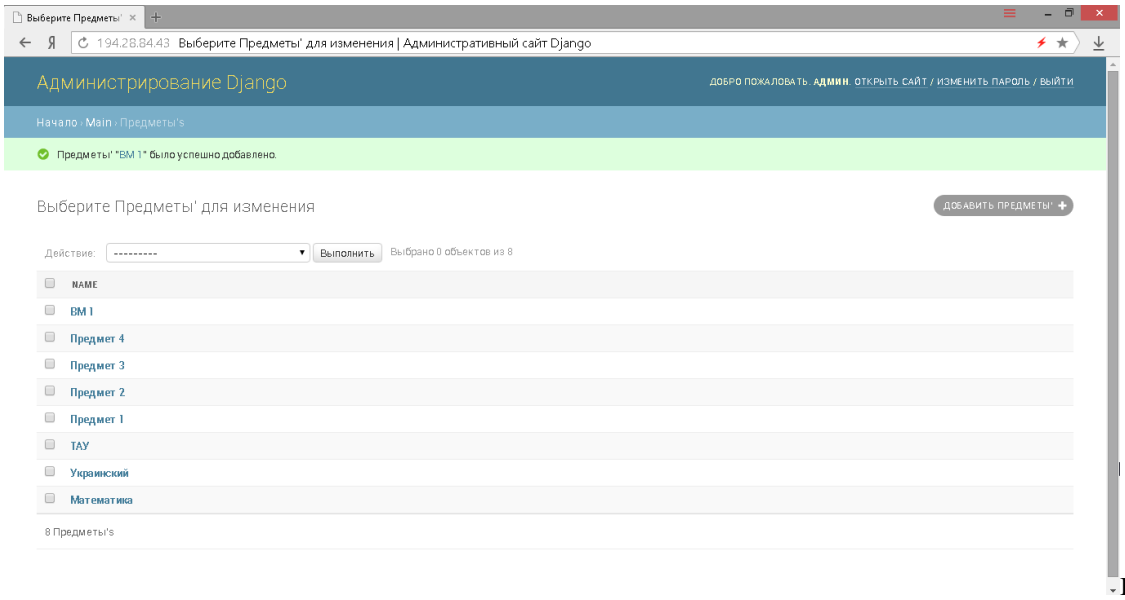


Рисунок 6.1.9 — Екран створення предмета



Ц

Рисунок 6.1.10 — Екран перегляду предметів

- результати тестів

Також функція для керування та виправлення помилок. Дозволяє додавати, редагувати та видаляти результати тестів.

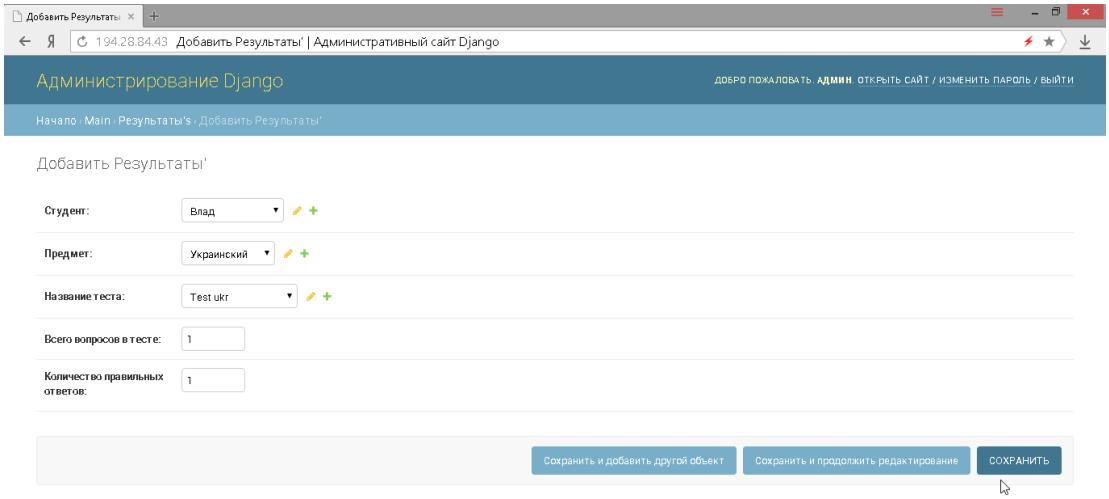


Рисунок 6.1.11 — Екран ручного додавання результату теста

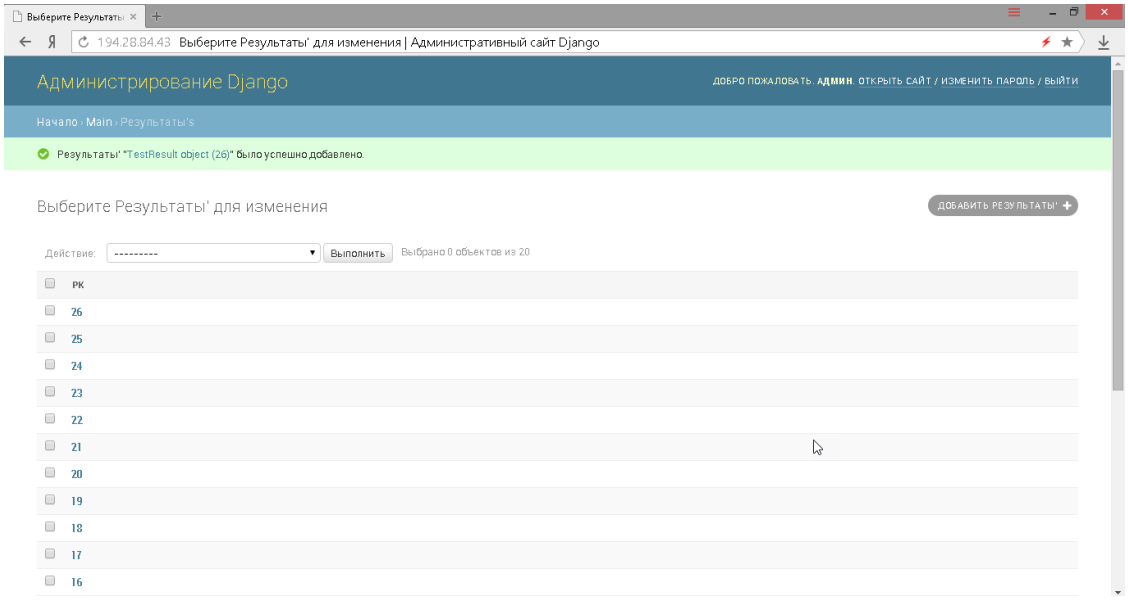


Рисунок 6.1.12 — Екран перегляду результатів тетстів

- додавати та змінювати параметри користувачів та давати користувачам необхідні права

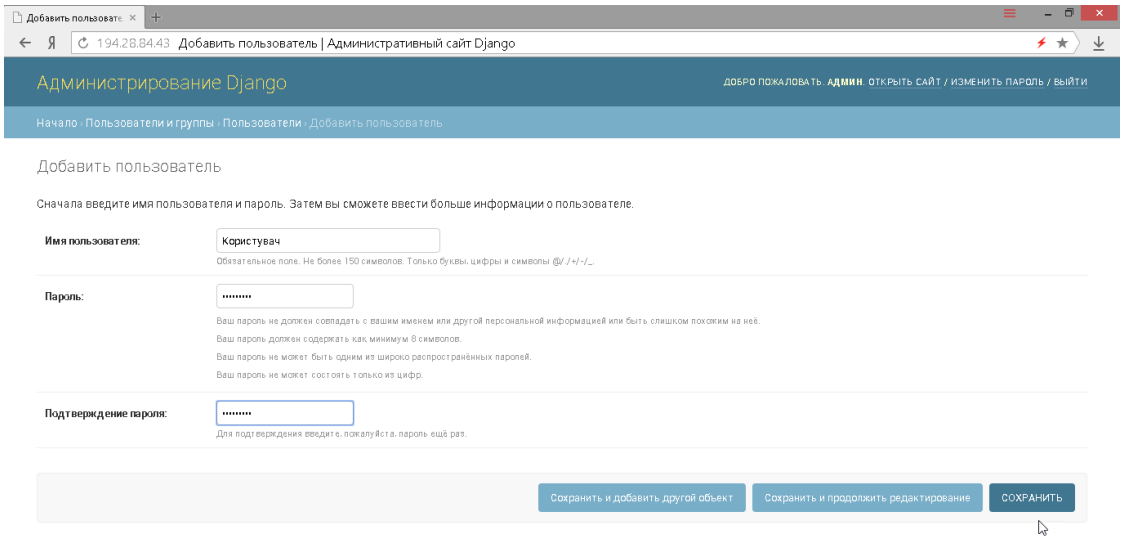


Рисунок 6.1.13 — Редагування параметрів користувача

Це одна з основних функцій адмін.-панелі кафедри. Надає можливість додавати нових користувачів. Основна ідея у створенні нових амінів для адмін-панель викладачів, тобто надавати викладачам можливість використовувати відповідну панель.

- створювати групи користувачів

Також важлива функція. Надає можливість при створювати групи користувачів по правам. Наприклад, ми хочемо надати можливість лаборанту додавати певні підручники і при цьому не мати змогу редагувати тести та їх результати… Щоб кожного разу при створенні користувача не обирати з цілого списку які ж права йому необхідні, можна просто один раз створити групу, давши їй назву та обравши необхідні поля і при наступному створенні нового користувача просто вказувати групу, тобто набір прав які йому необхідні.

Основною функцією адмін.-панелі кафедри є саме можливість створювати користувачів та надавати їм необхідні права.

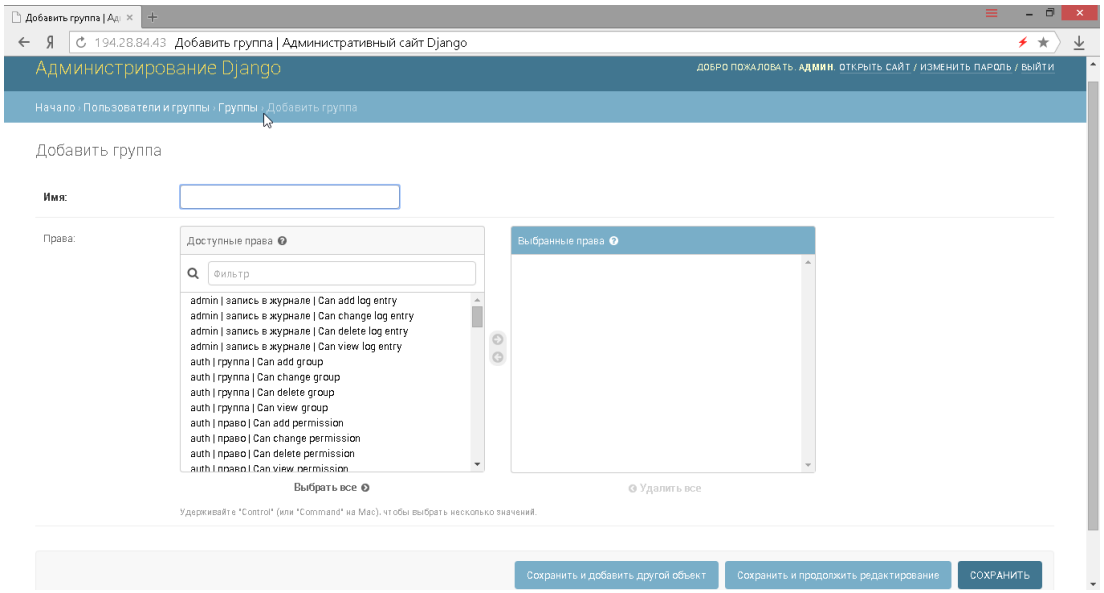


Рисунок 6.1.14 — Екран створення групи за правами

Найкращим варіантом є призначення однієї людини від кафедри що буде відповідати за цю панель.

6.2 Адмін-панель викладачів

Адмін-панель викладачів також реалізована як web-сторінка що надає змогу:

- переглядати список користувачів

Викладач може переглядати перелік судентів та вносити необхідні корективи.

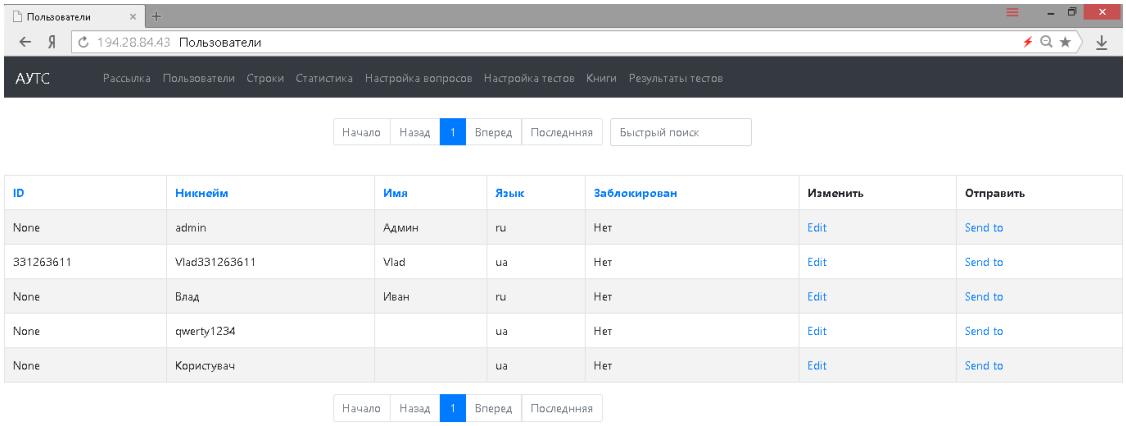


Рисунок 6.2.1 — Екран перегляду списку студентів

- проводити розсилку повідомлень

Викладачу дається можливість проводити розсилку — це може бути як повідомлення окремому користувачу, так і кожному. До повідомлення також можна додати файл. Це може бути певна новина, інструкція, користі матеріали або пропозиція пройти певний тест.

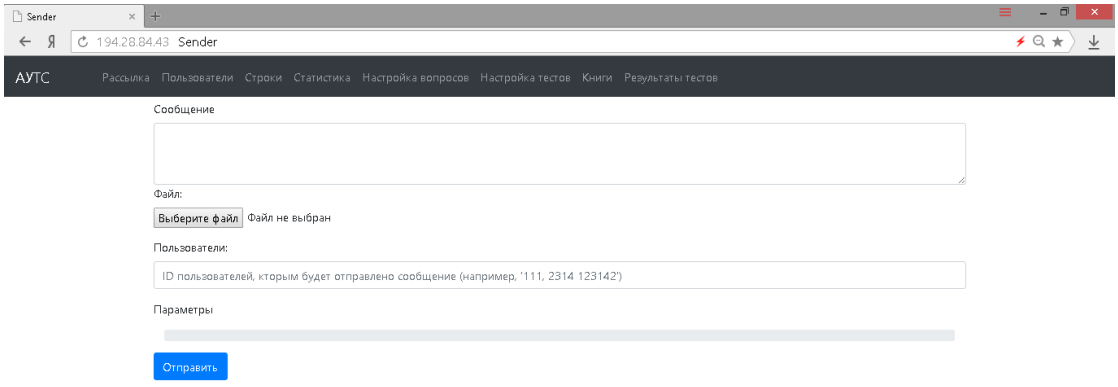


Рисунок 6.2.2 — Екран розсилки повідомлень

- переглядати та змінювати строки

На адмін-панелі викладача є можливість редагувати строки — це кістяк інтерфейсу чат-бота. Це базові сповіщення та кнопки що відображаються користувачу.

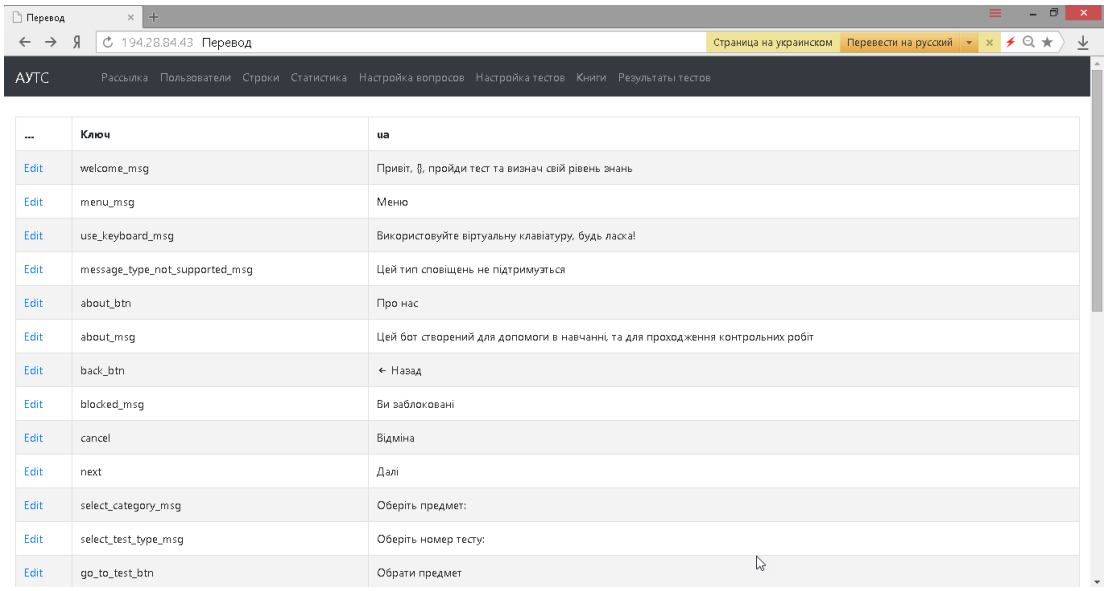


Рисунок 6.2.3 — Екран перегляду строк

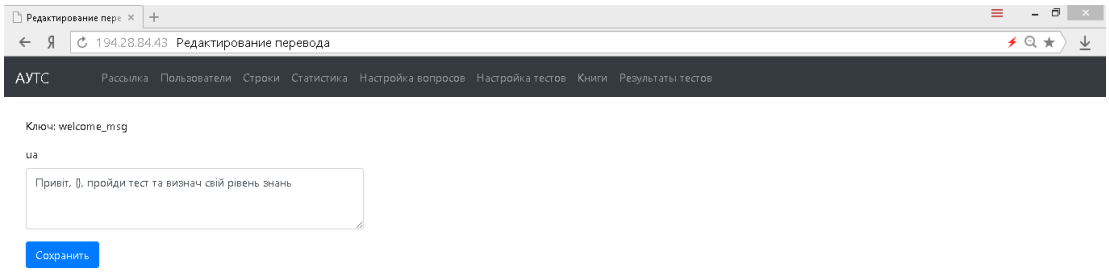


Рисунок 6.2.4 — Екран редагування строки

- переглядати статистику

Ця можливість дає змогу аналізувати навантаження на цього бота, зогальну кількість користувачів, кількість натискання кнопок, графік

користування ботом за останні 30 діб а також артефакти, що дозволяє відслідковувати збої.

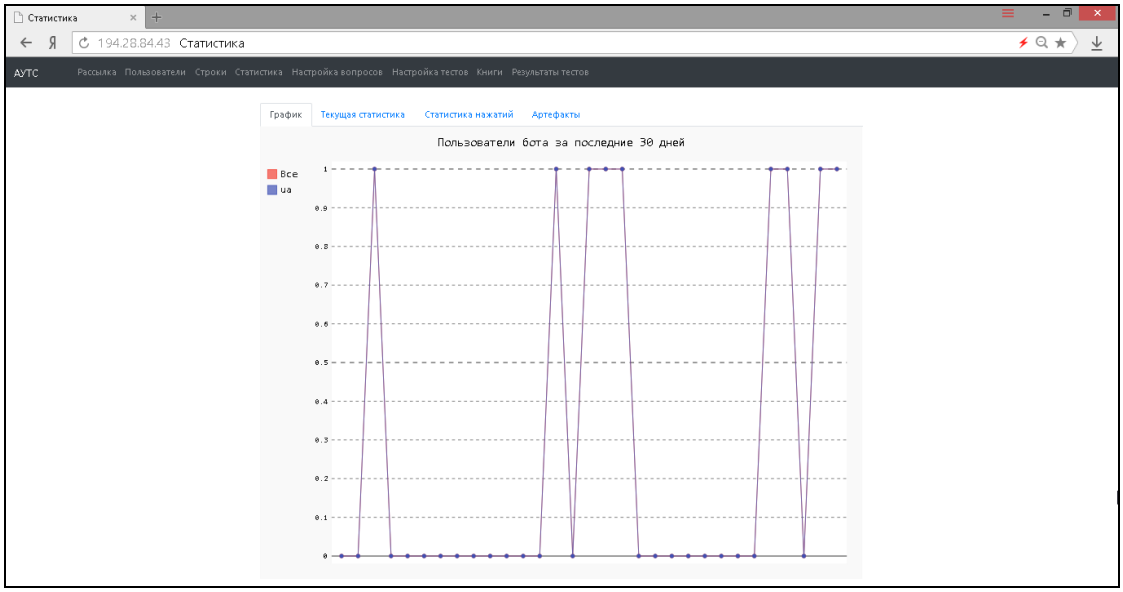


Рисунок 6.2.5 — Екран перегляду статистики

- проводити налаштування тестів

Ця функція дозволяє створювати та редагувати тести.

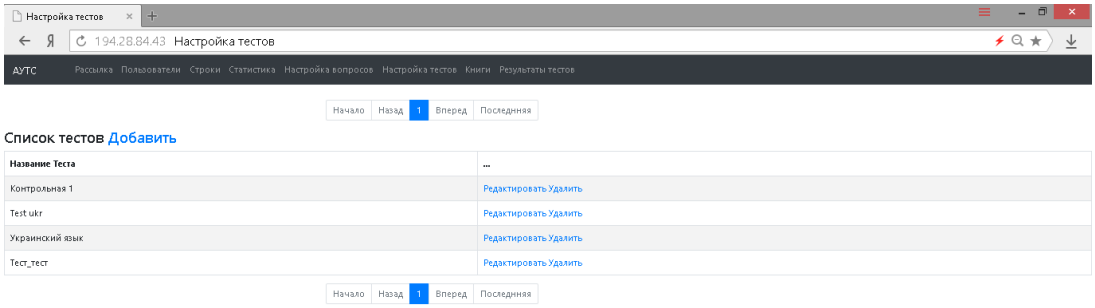


Рисунок 6.2.6 — Екран перегляду списку тестів

- проводити налаштування питань до тестів

Ця функція дозвояє створювати питання до тестів та зв’язувати їх, прив’язувати тести до предметів. Це одна з основних функцій цієї панелі.

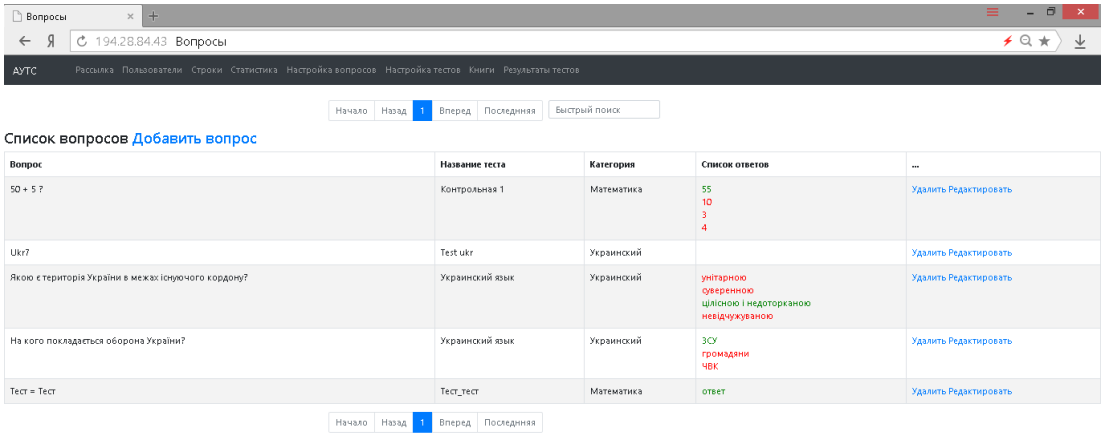


Рисунок 6.2.7 — Екран перегляду питань до тестів

- додаванн та редагування книг

Додатковою функцією є можливість додавати корисну літературу. Завдяки цій функції користувач може до певної дисципліни додати матеріал (посібник, підручник, методичку і тп) і при можливості дати посилання на його скачування. Ця функція також може полегшити життя студентів. Можна додавати матеріали самому, або через адмін-панель кафедри дати таку можливість старості або іншій відповідалній особі.

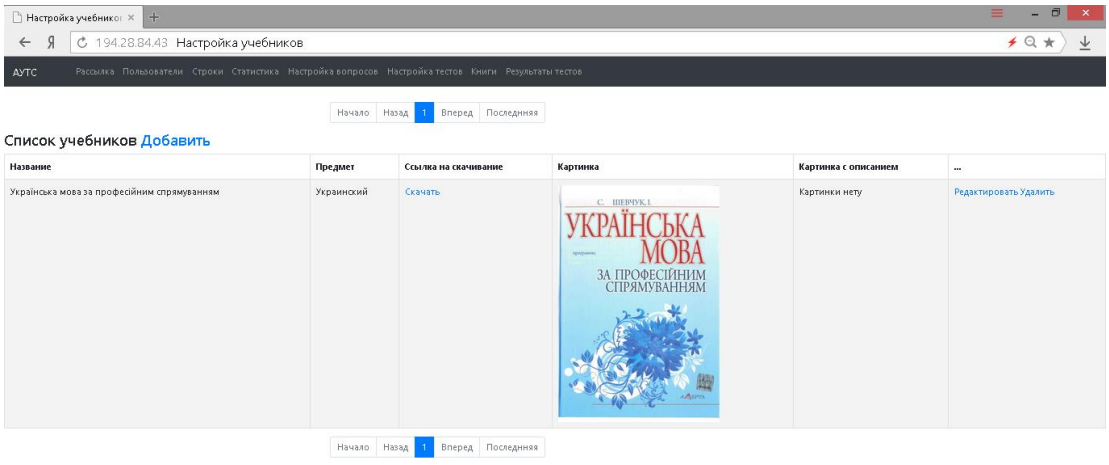


Рисунок 6.2.8 — Екран перегляду підручників

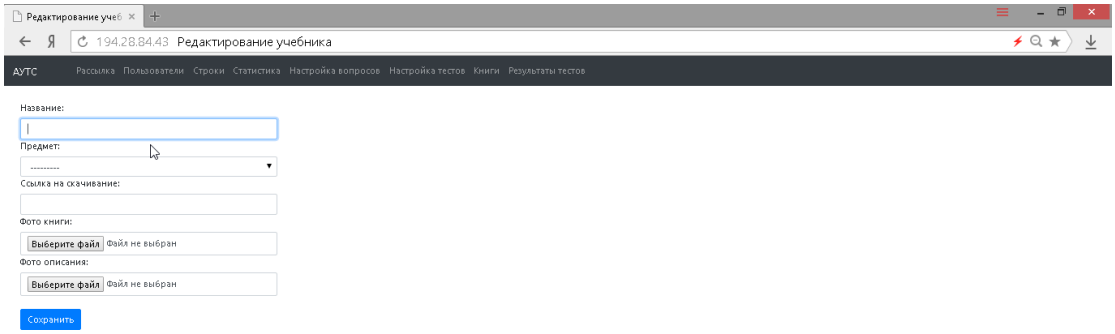


Рисунок 6.2.9 — Екран додавання підручників

- перегляд результатів тестів

Це також одна з основних функцій адмін-панелі, вона дає можливість переглядати результати нещодавно пройдених тестів.

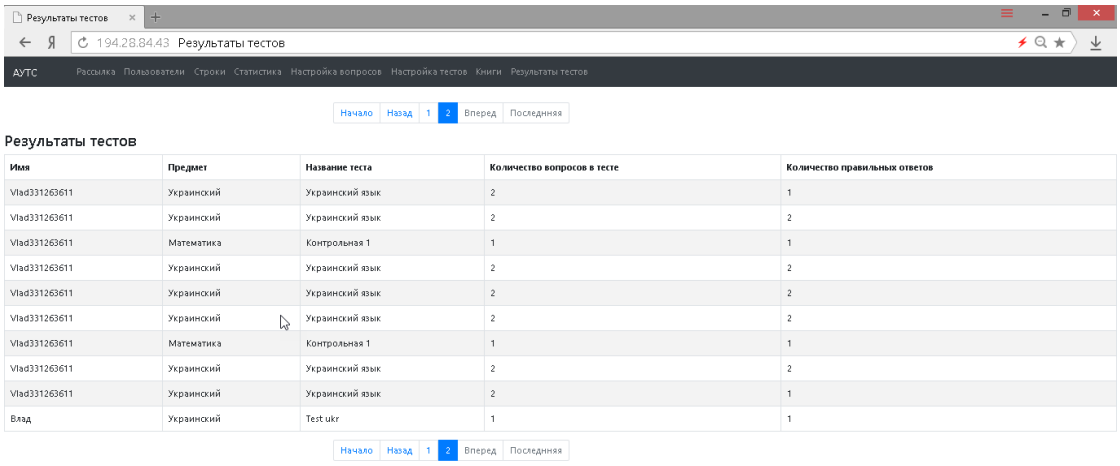


Рисунок 6.2.10 — Екран перегляду результатів тестувань

6.3 Чат-бот для студентів

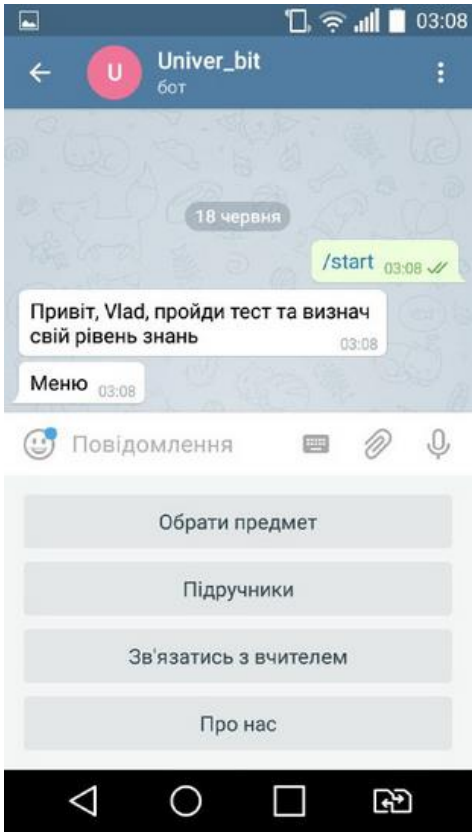


Рисунок 6.3.1 — Початковий екран бота

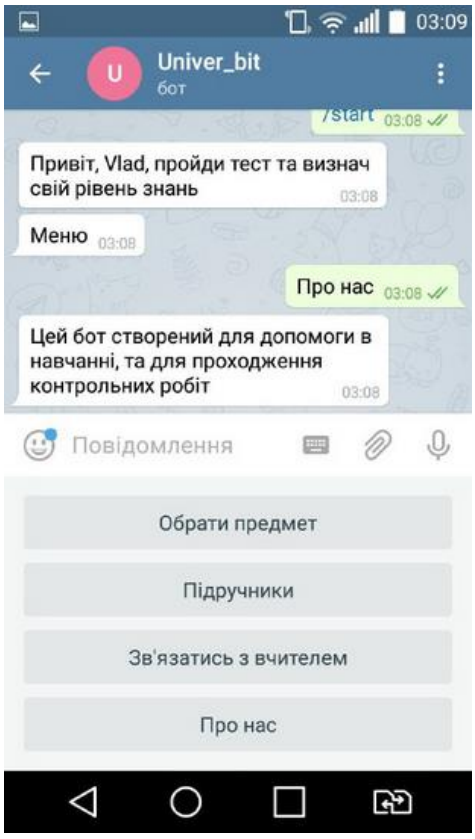


Рисунок 6.3.2 — Повідомлення «Про нас»

Чат-бот являється основою цього проекту, він поєднує у собі простоту у використанні та невеликий, але дуже корисний перелік можливостей, а саме:

- проходження тестів

Основною функцією чат-бота є можливість проходження тестів. Для проходження потрібного тесту студент обирає предмет а потім обирає необхідний тест. Студенту одразу відображається кількість правильних відповідей.

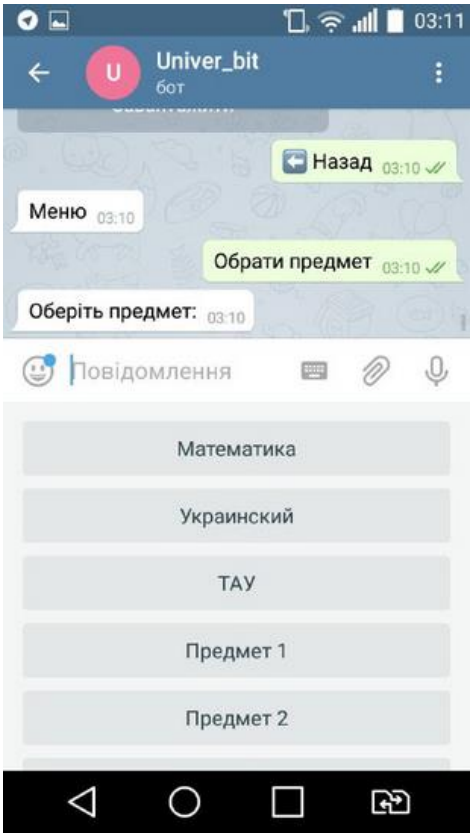


Рисунок 6.3.3 — Вибір предмета

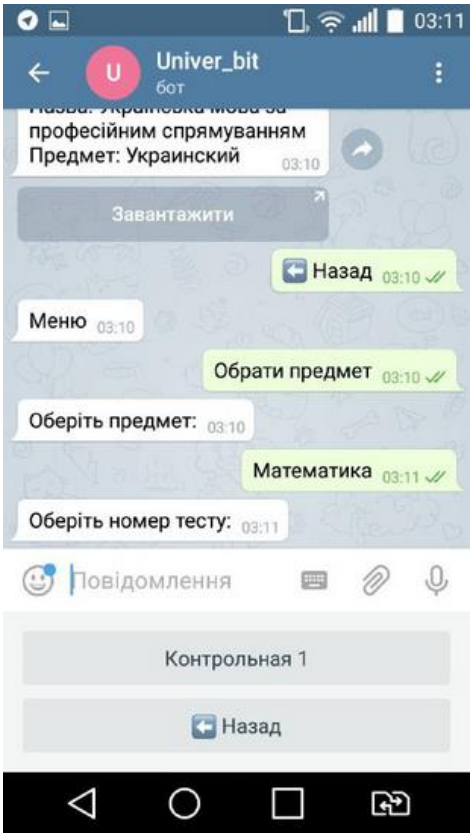


Рисунок 6.3.4 — Вибір теста

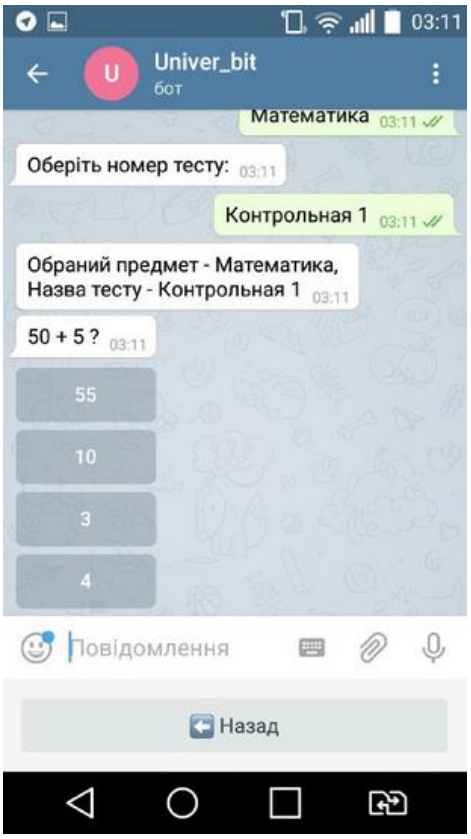


Рисунок 6.3.5 — Питання та варіанти відповідей

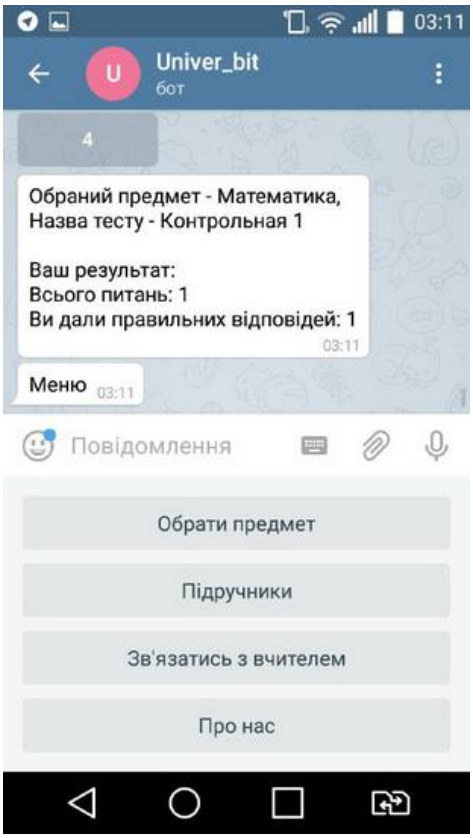


Рисунок 6.3.6 — Результати проходження тесту

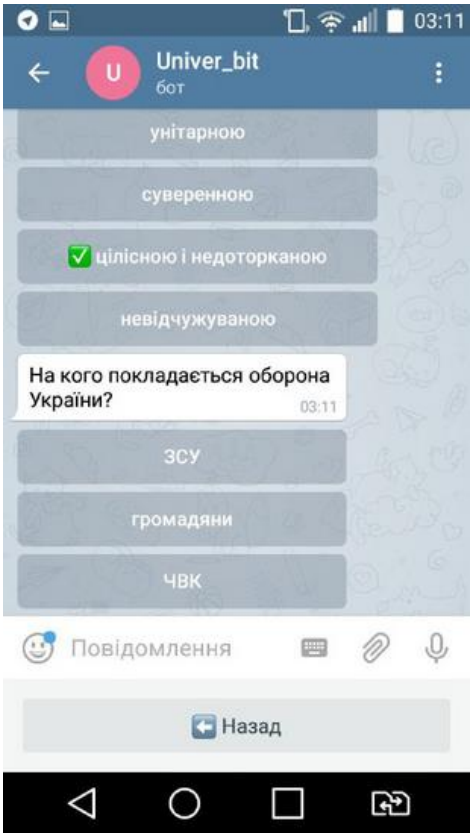


Рисунок 6.3.7 — Правильна відповідь

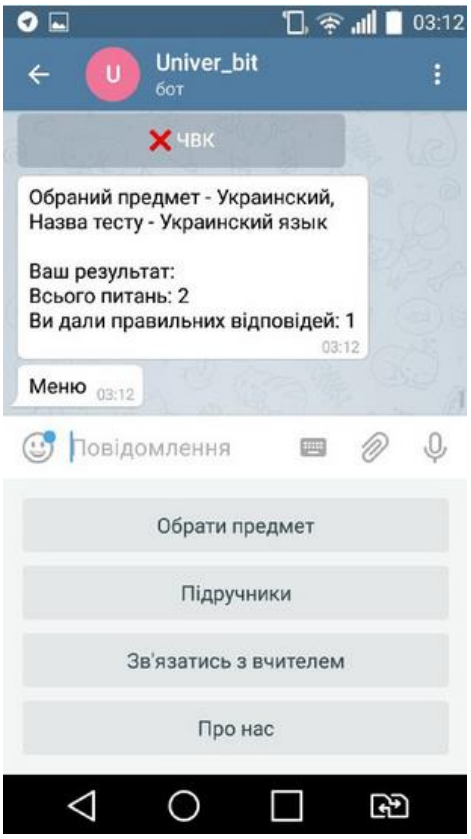


Рисунок 6.3.8 — Неправильна відповідь

- перегляд та скачування необхідної літератури

Це додаткова але дуже корисна функція. Студент може переглядати та скачувати навчальні матеріали, якщо вои були додані викладачем або іншою відповідальною особою.

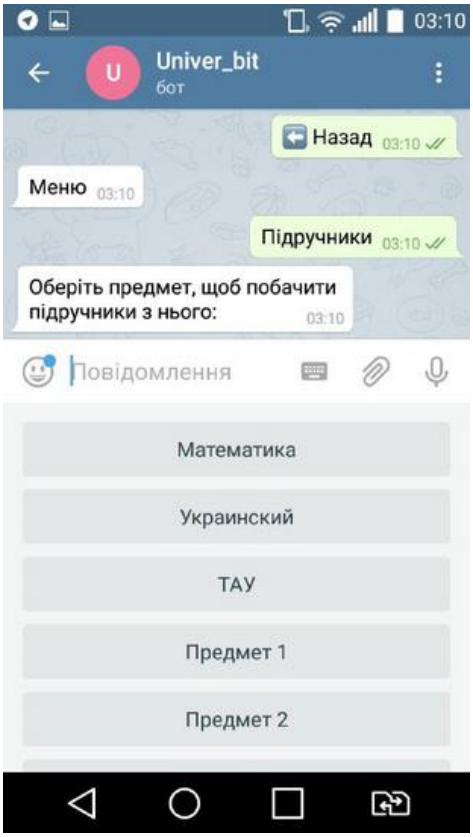


Рисунок 6.3.9 — Вибір предмета для перегляду підручників

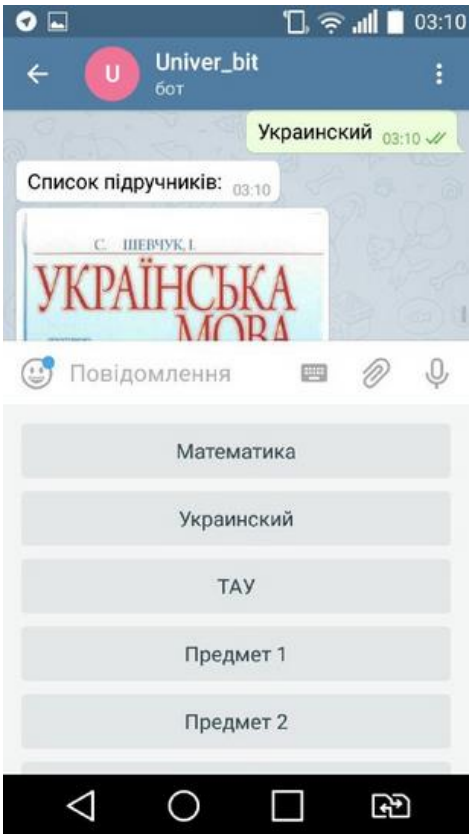


Рисунок 6.3.10 — Перегляд підручників

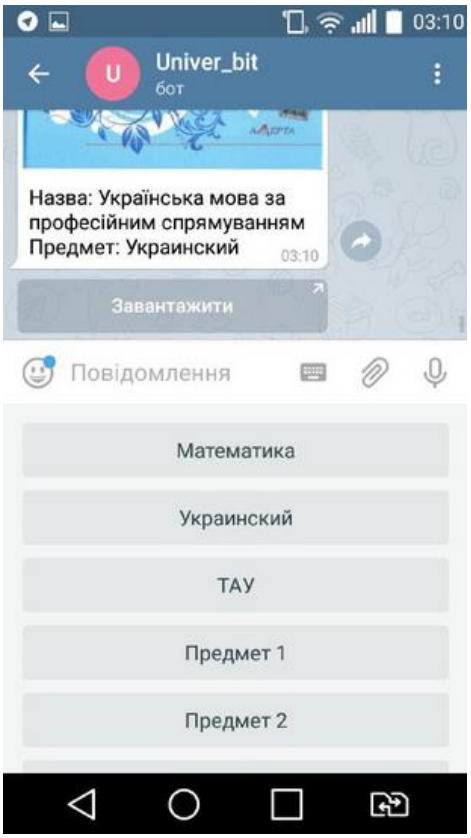


Рисунок 6.3.11 — Посилання на завантаження підручника

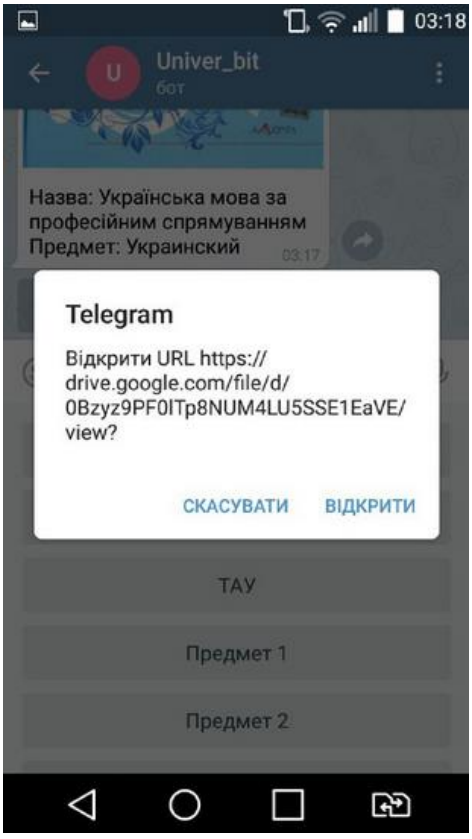


Рисунок 6.3.12 — Перехід по посиланню для завантаження підручника

- зв’язок з викладачем

Ще одна додаткова і корисна функція — користувач може написати повідомлення для отримання зворотнього зв’язку. Це може бути пропозиція, побажання, запитання, скарга і тп. Це будь-яке повідомлення. Воно прийде у telegram-канал і на нього, при необхідності, зможе відповісти відповідальна особа.

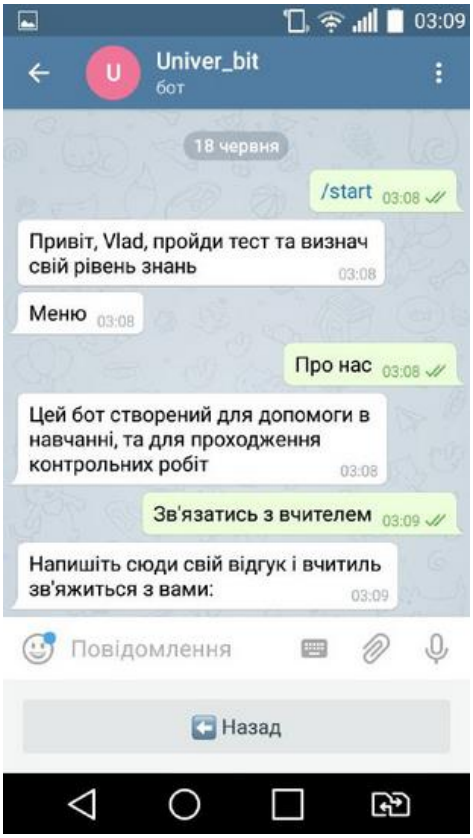


Рисунок 6.3.13 — Натискання кнопки «Зв’язатися з вчителем»

Рисунок 6.3.14 — Відправлення повідомлення в Telegram-канал

6.4 Telegram-канал

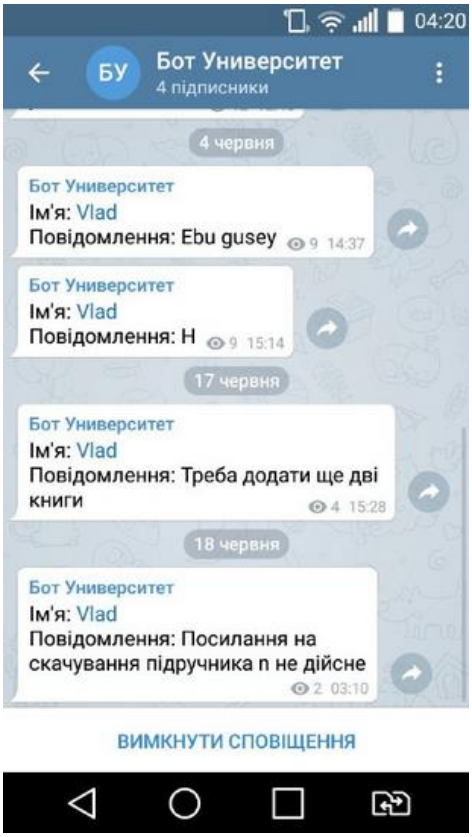


Рисунок 6.4.1 — Telegram-канал

Основною функцією telegram-каналу є отримання зворотнього зв’язку від студентів.У telegram-канал приходять усі повідомлення, які студенти відправляють з чат-боту. Разом з повідом відображається посилання на того, хто його залишив, що спрощує комунікацію. Ідея заключається у тому, що студент може залишити побажання, пропозицію або скаргу і відповідальна людина дасть зворотній зв’язок. Це набагато зручніше ніж група, оскільки не виникає флуду і студент не соромиться своїх однолітків і може задавати будб-які питання щодо освітнього процесу або пропонувати певні покращення. Боротися з флудом дуже легко — можна видалити або заблокувати порушника через адмін.-панель кафедри. Єдиним мінусом такого рішення є те, що викладач дає відповідь лише тому хто задав запитання і тп. Але ця проблема вирішується завдяки розсилці на адмін.-панелі викладача. Рекомендованим способом використання каналу є назначення одного відповідального.

7 Тестування

Мануальне тестування проводилося упродовж всієї розробки, і її результат можна побачити у розділі № 5 — «Інструкція користувача». Для описання інтерфейсів вони усі були повторно пройдені та переглянути. Немає сенсу знов проводити ручне тестування, придумувати кейси і сценарії. Усі критичні шляхи були пройдені успішно і тому можна вважати, що користувацька частина та інтерфейси працюють коректно. Але чи можна вважати, що система була повністю протестована і вважати її повністю справною? Звісно ж ні, оскільки баги та дефекти є у кожній системі, їх не можливо повністю і одразу покрити як тестами так і фіксаме. Але чи були покриті тестами усі основні вимоги?

Однією з основних вимог до подібної системи є стійкість до навантаження, оскільки нею буде користуватись одразу велика кількість студентів. У цьому розділі буде проаналізована робота системи під різними навантаженнями, а саме порівняємо роботу системи при тридцяти, ста двадцяти та чотирьох тисячах користувачів. Це імітує поведінку системи при проходженні тестів студентами однієї групи, одного потоку та наприклад якогось факультетського опитування відповідно.

Тестувати сам бот на навантаження немає сенсу, оскільки є мікросервіс, який відповідає за отримання повідомлень, який не дозволить впасти самій системі. Основним слабким місцем подібних систем є навантаження — більшість простих проектів можна «положити» усього одним десятком користувачів. У цьому проекті дану проблему було вирішено на архітектурному рівні — було створено окремій мікросервіс,

Тестування проводиться за допомогою такого інструменту як Gatling. Gatling — це фреймвор для Scala, який створено спеціально для проведення тестування навантаження.Sender, який повністю відповідає за відправку повідомлень. Він налаштований так, що усі повідомлення йдуть до нього, і

навіть при неймовірних навантаженнях сама система буде «стояти». Більшість подібних проблем вирішується шляхом створення черги а також приорітизацією повідомлень.

Тестом слугуватиме простий скріпт, який створено за допомогою рекордера, тобто, було просто записано необхідну послідовність дій а потім у неї вносилися певні зміни.

Для проведення тестування навантаження було створено простий сценарій, у якому користувач перейшов на панель керування та послідовно переглядає усі сторінки. Далі змінюючи усього лише одну строку у скрипті було зімітовано необхідну кількість користувачів.

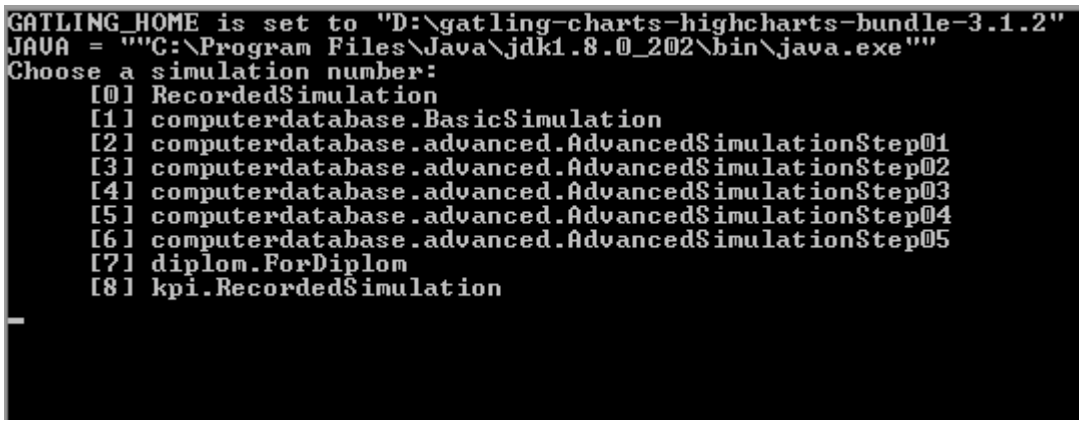


Рисунок 7.1

```
setUp(scn.inject(atOnceUsers(1))).protocols(httpProtocol)
```

Рисунок 7.2

Рисунок 7.3

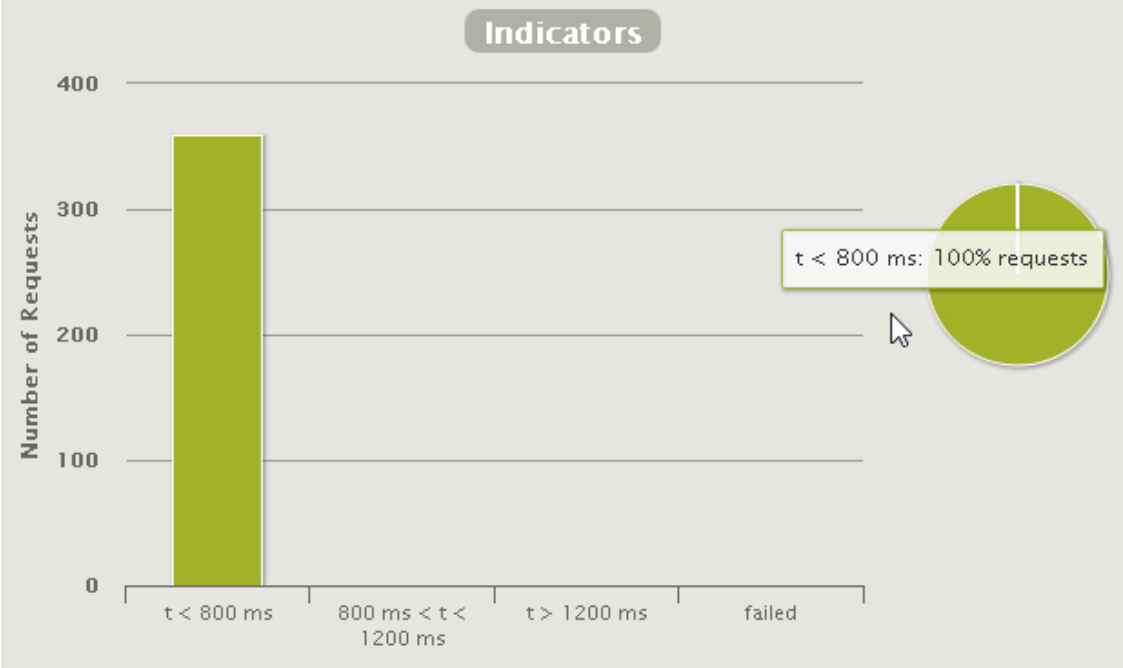


Рисунок 7.4

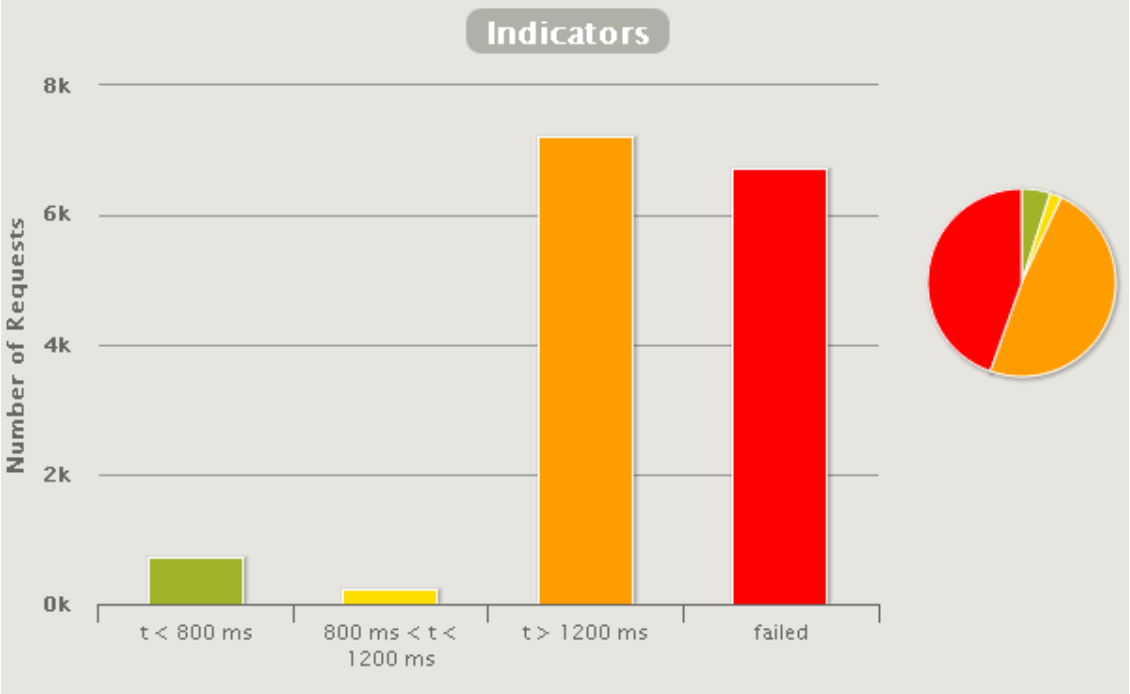


Рисунок 7.5

STATISTICS

Requests	Executions				
	Total	OK	KO	% KO	Req/s
Global Information	90	90	0	0%	2.812
request_0	30	30	0	0%	0.938
request_2	30	30	0	0%	0.938
request_3	30	30	0	0%	0.938

Рисунок 7.6

STATISTICS

Requests	Executions				
	Total	OK	KO	% KO	Req/s
Global Information	360	360	0	0%	10.909
request_0	120	120	0	0%	3.636
request_2	120	120	0	0%	3.636
request_3	120	120	0	0%	3.636

Рисунок 7.7

STATISTICS

Requests	Executions				
	Total	OK	KO	% KO	Req/s
Global Information	14907	8195	6712	45%	109.61
request_0	4000	4000	0	0%	29.412
request_2	4000	1093	2907	73%	29.412
request_3	4000	1906	2094	52%	29.412
request_4	2907	1196	1711	59%	21.375

Рисунок 7.8

Expand all groups | Collapse all groups

Response Time (ms)							
Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
68	143	268	321	348	353	181	89
157	293	301	342	351	353	284	41
68	87	97	103	104	104	87	11
132	143	231	236	237	237	172	46

Рисунок 7.9

Expand all groups | Collapse all groups

Response Time (ms)

Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
55	156	501	701	742	755	290	227
205	624	678	734	749	755	590	115
55	101	127	136	163	174	98	31
134	156	242	265	271	273	183	48

Рисунок 7.10

Expand all groups | Collapse all groups

Response Time (ms)

Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
0	10001	14299	40005	40387	60002	11412	9802
378	6555	7591	8340	8678	9907	6579	1295
59	11825	14125	16071	16317	22120	9900	5557
0	20471	39670	40232	40725	52432	20493	14041
50	10001	10058	10184	10264	60002	7650	3706

Рисунок 7.11

ERRORS

Error	Count	Percentage
i.n.c.ConnectTimeoutException: connection timed out: ocsp.pki.goog/172.217.20.163:80	4612	68.713 %
j.n.UnknownHostException: ocsp.sca1b.amazontrust.com	1093	16.284 %
i.n.c.ConnectTimeoutException: connection timed out: ocsp.sca1b.amazontrust.com/54.192.230.42:80	269	4.008 %
i.n.c.ConnectTimeoutException: connection timed out: ocsp.sca1b.amazontrust.com/54.192.230.139:80	250	3.725 %
i.n.c.ConnectTimeoutException: connection timed out: ocsp.sca1b.amazontrust.com/54.192.230.10:80	242	3.605 %
i.n.c.ConnectTimeoutException: connection timed out: ocsp.sca1b.amazontrust.com/54.192.230.193:80	240	3.576 %
j.i.IOException: Premature close	5	0.074 %
i.g.h.c.i.RequestTimeoutException: Request timeout to ocsp.pki.goog/172.217.20.163:80 after 60000 ms	1	0.015 %

Рисунок 7.12

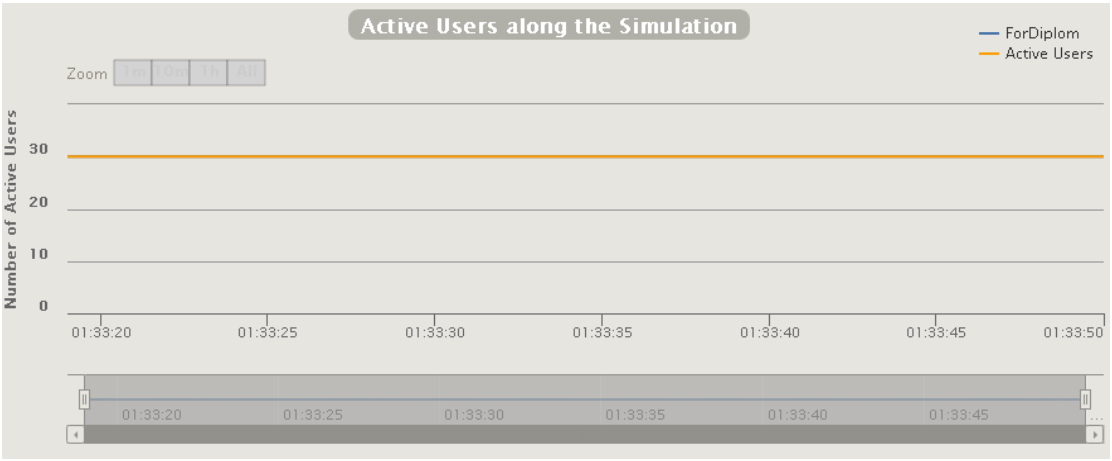


Рисунок 7.13

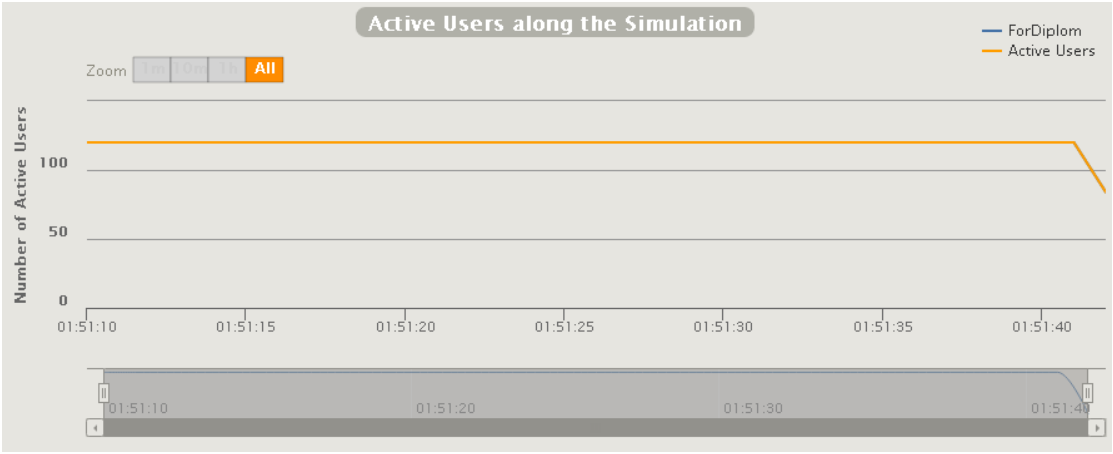


Рисунок 7.14

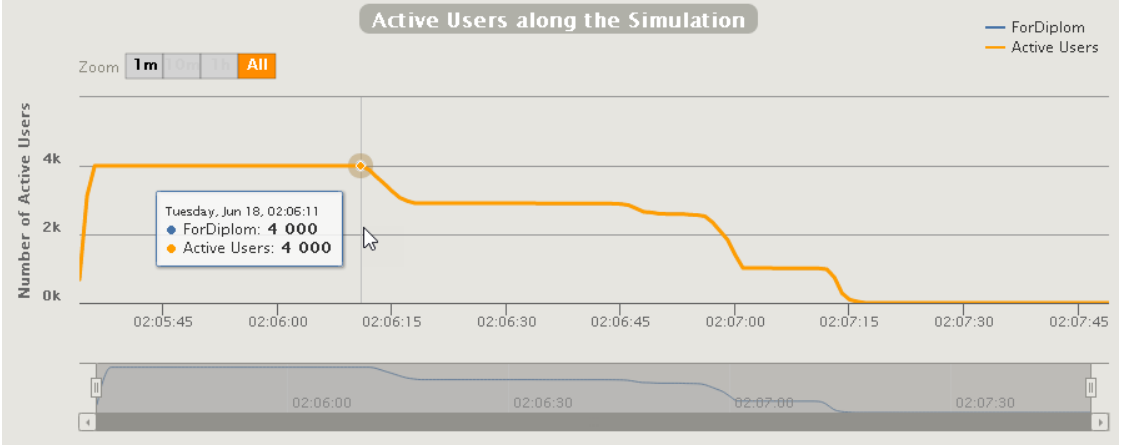


Рисунок 7.15

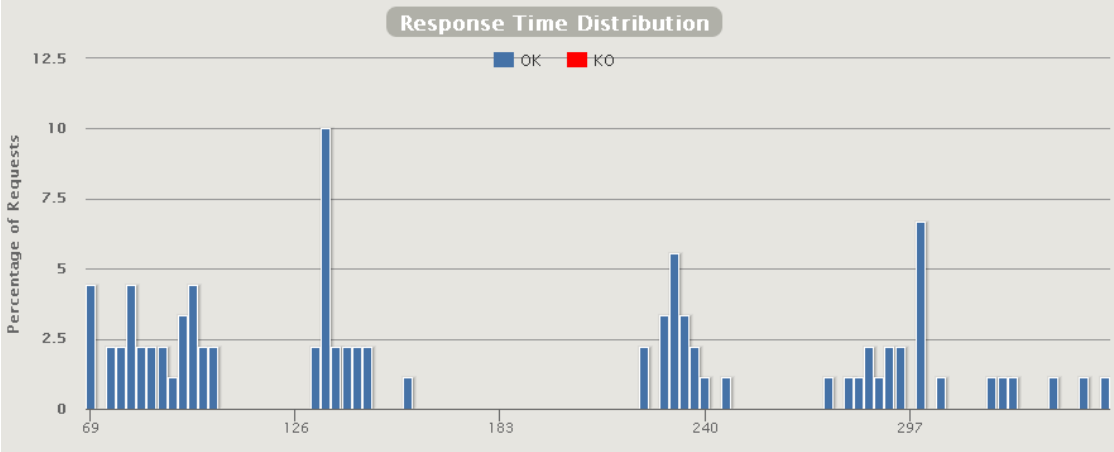


Рисунок 7.16

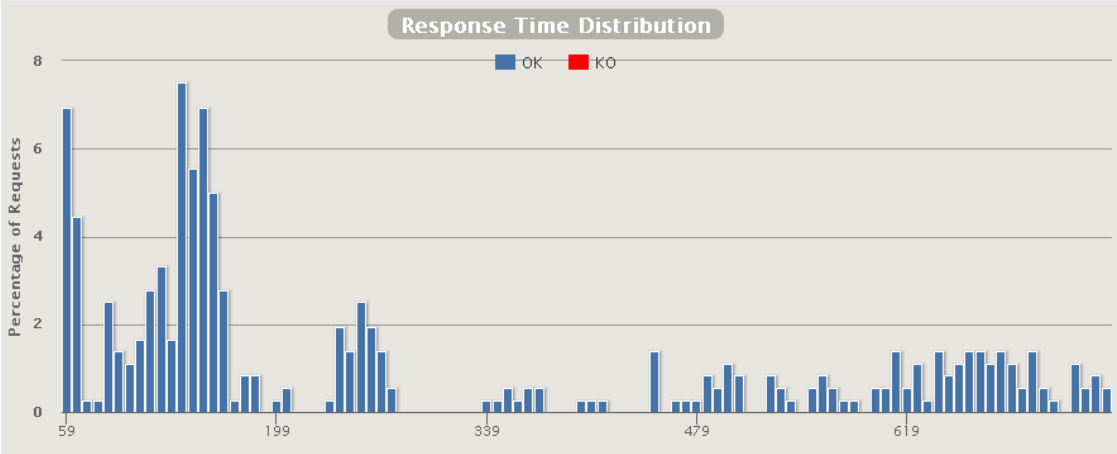


Рисунок 7.17

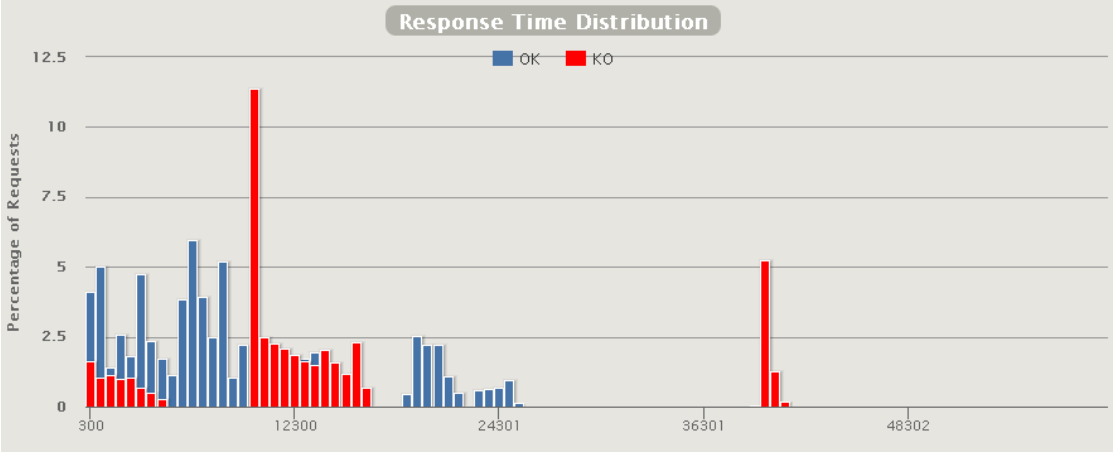


Рисунок 7.18

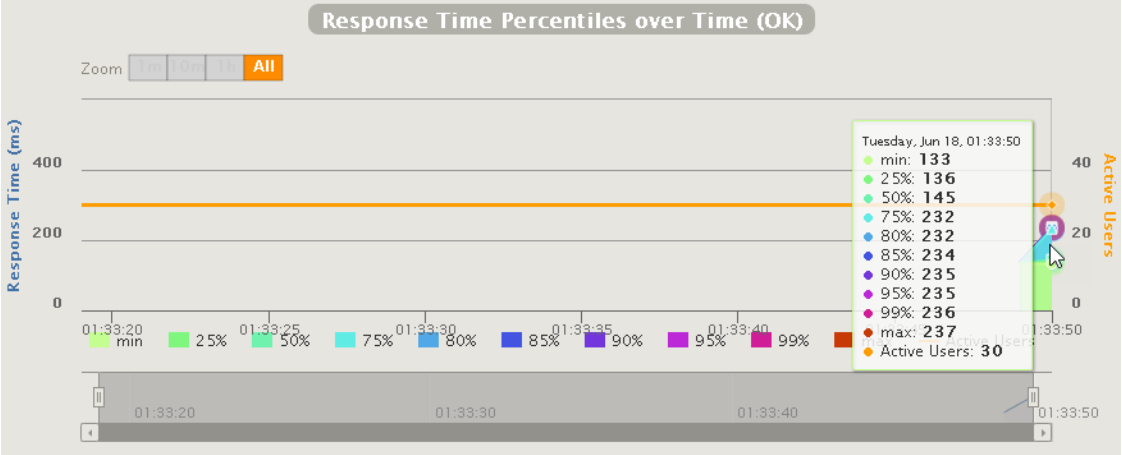


Рисунок 7.19

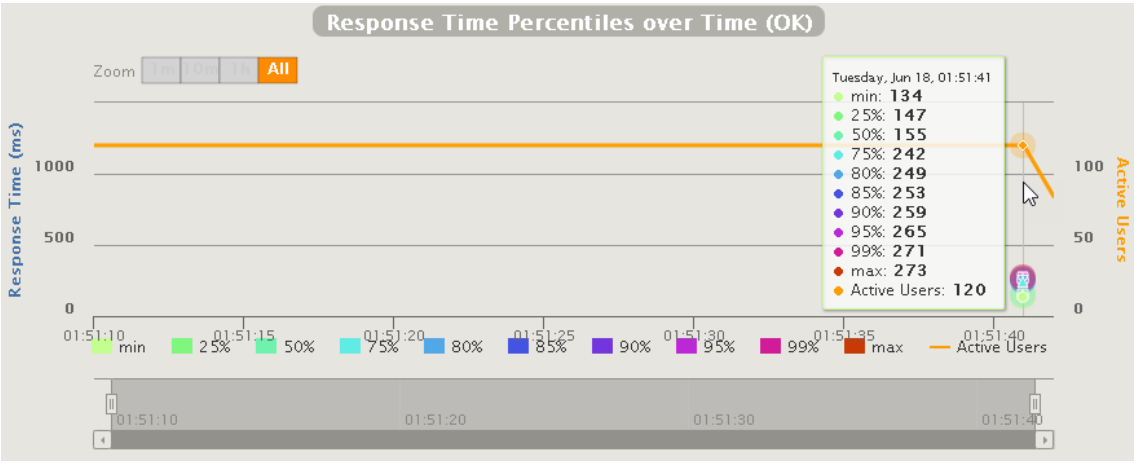


Рисунок 7.20

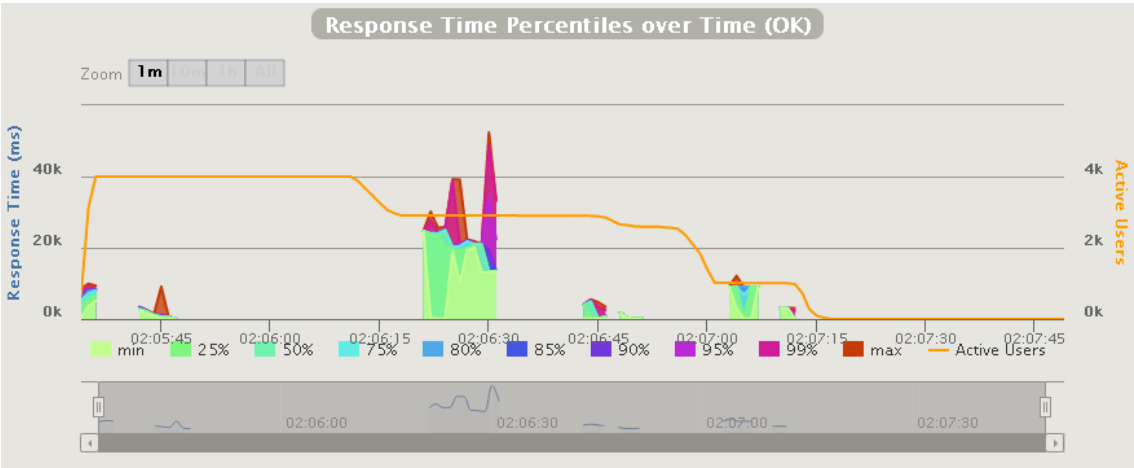


Рисунок 7.21

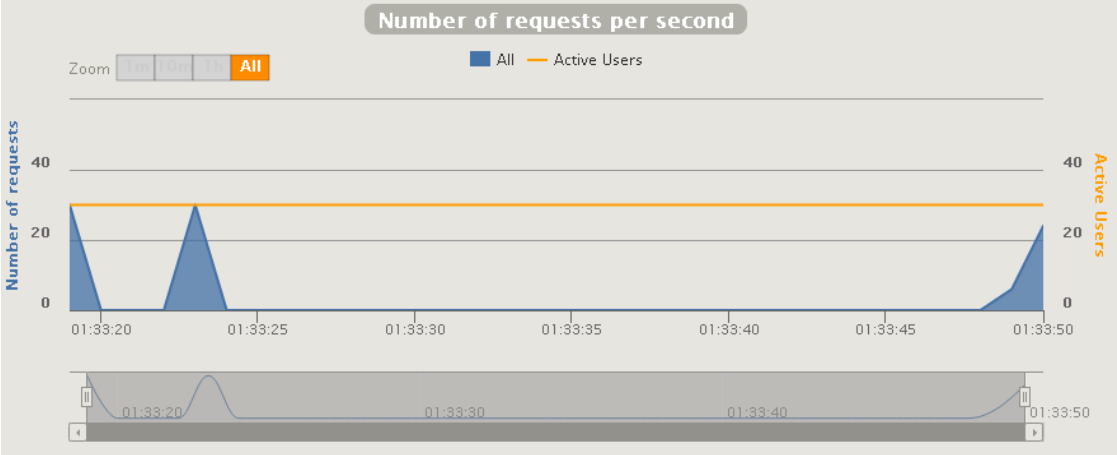


Рисунок 7.22

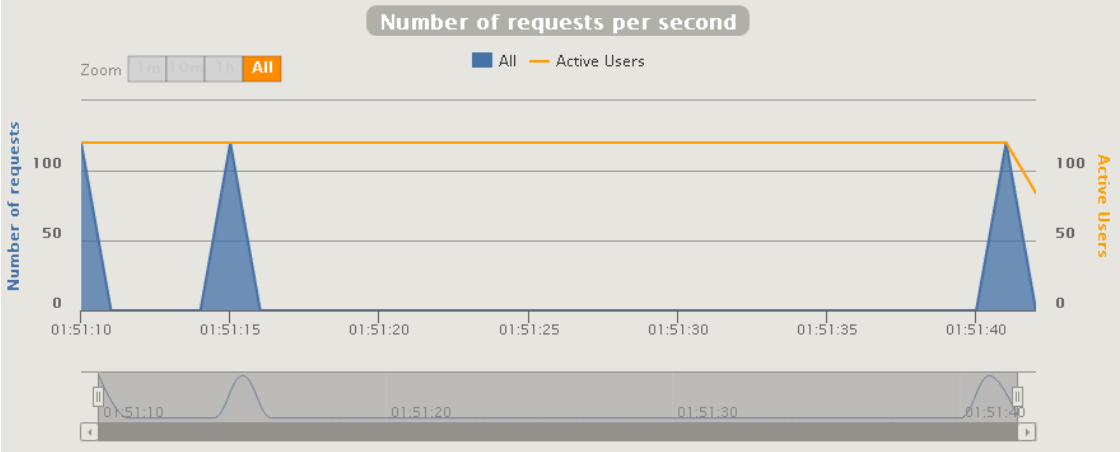


Рисунок 7.23

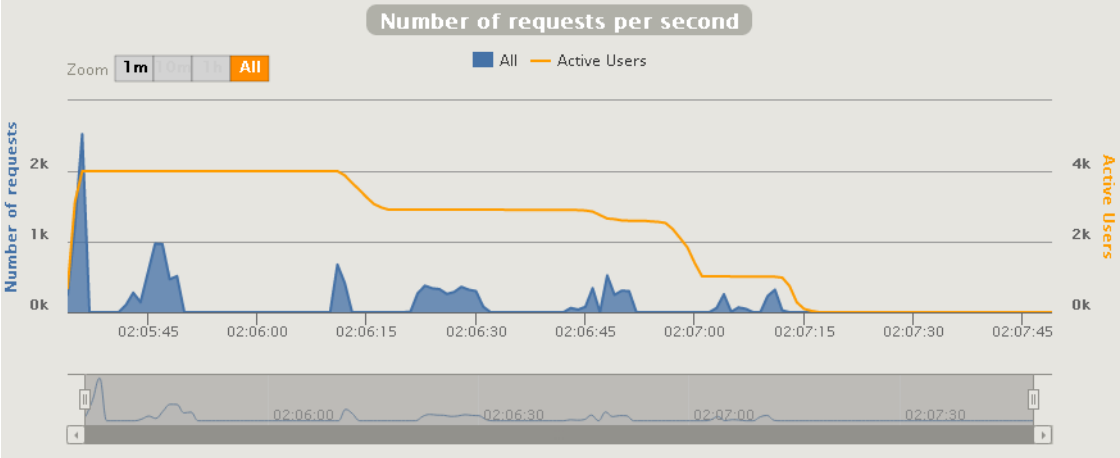


Рисунок 7.24

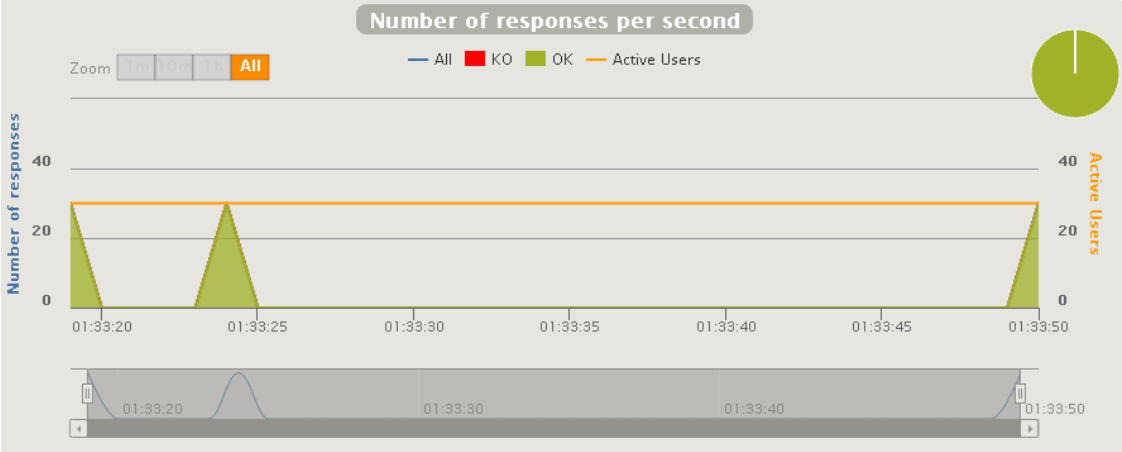


Рисунок 7.25

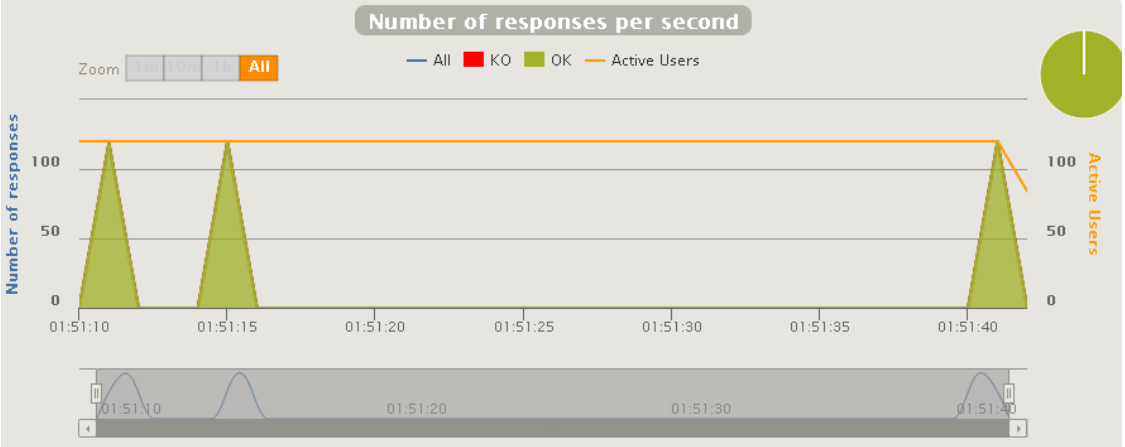


Рисунок 7.26

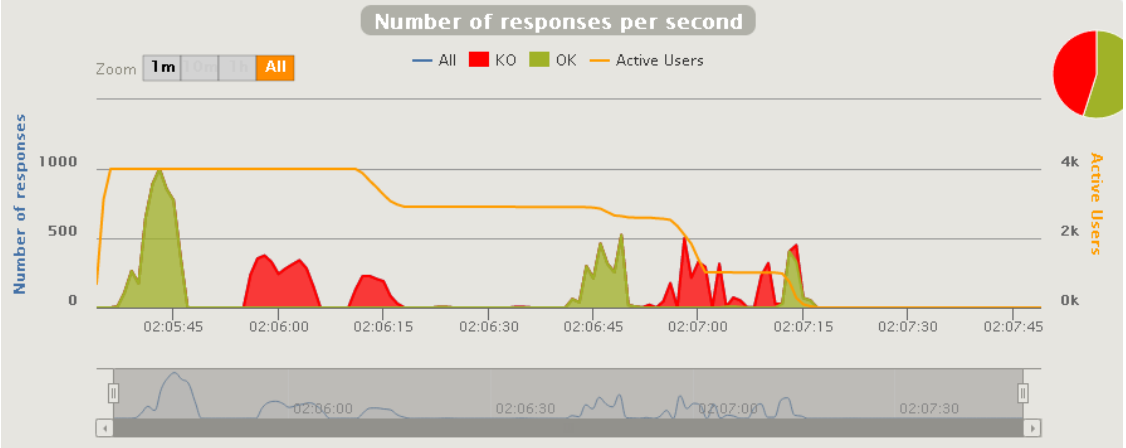


Рисунок 7.27

Висновок до розділу № 7

Отже було проведено повноцінне тестування данної системи як в ручну, так і за допомогою спеціальних інструментів. З отриманих результатів можна зробити висновок, що система повністю задавольняє поставлені вимоги.

Висновок

Відповідно до завдання на курсовий проект.було розроблено систему, що складається з адмін-панель кафедри, адмін-панель викладачів, чат-бот для студентів та telegram-канала, повністю задовольняє поставлені вимоги. Було проведено аналіз існуючих рішень та обрано оптимальний варіант структури системи, були прийняті рішення, що сильно підвищили «живучість» усієї системи. Також був проведений аналіз існуючих інструментів та технологій та обрано найбільш підходящий набор.

Однією з основних частин даного проекту стало проведення автоматизованого тестування для підтвердження ефективності та стійкості системи. У результаті цього можна упевнено сказати, що було отримане якісне рішення поставлених задач.

Список літератури

1. Електронний кампус - https://ecampus.kpi.ua/about.
2. Стаття про Тесторіумй - http://www.znanius.com/5520.html.
3. Офіційний сайт INDIGO - https://indigotech.ru/.
4. Стаття “10 причин, почему в веб-разработке Pethon выигрывает у PHP” - https://medium.com/futureinapps/10-причин-почему-в-веб-разработке-python-выигрывает-у-php-6e16d1d73b4b.
5. Стаття “PYTHON 2 VS PYTHON 3” - https://www.8host.com/blog/python-2-vs-python-3-kratkij-obzor-i-prakticheskie-soobrazheniya/.
6. Стаття “Django – фреймворк на Python” - https://web-creator.ru/articles/django.
7. Стаття “Плюсы и минусы Django ” - https://python-scripts.com/django-obzor.
8. Стаття “Flask против Django: почему Flask может быть лучше” - https://python-scripts.com/flask-vs-django.
9. Стаття “SQLite – замечательная встраеваемая БД (часть 1)” -. https://habr.com/ru/post/149356/.
10. Стаття “HTML сайт. Преимущества и недостатки HTML сайтов” -. http://inoption.info/sitehtml.html.
11. Стаття “Что такое JavaScript?” - https://www.ipipe.ru/info/javascript.html.
12. Конспект лекцій з курсу "WEB-програмування".
13. Конспект лекцій з курсу "Сучасні технології програмування".